Sistema de Control de Aforo en Espacios Cerrados

Por

Víctor Tello Carrascal

Jesús Álvarez Coll

Miguel Artell Moreno

Daniel Grado Guerrero

UNIVERSIDAD COMPLUTENSE MADRID

Trabajo de fin de grado del

Grado en Ingeniería Informática / de Computadores

Facultad de Informática

Dirigido por

Sergio Bernabé García

Guillermo Botella Juan

Occupancy-based Control System for Closed Spaces

Madrid, 2020–2021

Índice general

Índice de figuras

Índice de tablas

Resumen

A día de hoy la población humana se ve en un creciente aumento, esto junto a la situación actual de pandemia propiciada por la COVID-19, ha hecho que sea necesario aumentar los controles de aforo en los diferentes espacios ya sean públicos o privados.

Hasta ahora, se han visto diferentes sistemas que permiten controlar la cantidad de personas que hay en un espacio, desde un control manual con una persona en la puerta encargada de contabilizar las entradas y salidas, hasta aplicaciones complejas que se encargan de contar automáticamente.

En este proyecto se va a proporcionar una solución de bajo coste que sea capaz de cumplir este objetivo. Para ello, se van a emplear diferentes opciones que otorgan una medida más o menos precisa. Las soluciones se basan en el uso de sensores infrarrojos o de sensores de ultrasonido, que si bien, nos permiten controlar la cantidad de personas que entran y salen, el resultado que nos dan es aproximado, y en el uso de inteligencia artificial, para a través de una cámara que capta imágenes en el momento, poder detectar a las personas y hacer un seguimiento a lo largo del entorno. Todos los datos recopilados se enviarán en tiempo real mediante el uso del Internet of Things (IoT) y llegarán a una aplicación web que servirá como interfaz de usuario para que estos puedan acceder a los datos de una manera más amigable.

Palabras clave

Control de ocupación, tiempo real, IoT, aplicación web, sensores, visión por computador.

Abstract

Nowadays, the human population is suffering an increasement, alongside the actual situation caused by the COVID-19 pandemic, has made it necessary to increase the occupancy controls in different places, wether public or private.

Untill now, different systems that allow controlling the number of people inside an enclosure have been seen, from a manual control with someone at the door in charge of counting the entrances and exits of people, to complex applications that automate this task.

The aim of this project is to provide a low-cost solution capable of accomplishing this objective. To make it possible, different options will be used in order to supply a nearly precise measurement. This solutions are based on the use of sensors such as infrared or ultrasonic, wich although allow us to control the people entering and leaving, their results are approximate; while using an artificial inteligence, through a camera that caputures real time images, is able to detect and track people throughout the environment. All collected data will be sent in real time using IoT, and will arrive to a web application that will serve as a user interface, so they could be accessed in a friendlier way.

Keywords

Occupancy control, real-time, IoT, web application, sensors, computer vision.

Capítulo 1

Introducción

1.1. Motivación

A lo largo del último siglo la población humana se ha incrementado de manera exponencial, y se espera que siga creciendo aún más. Todo esto contribuye a la existencia de eventos multitudinarios, a la gran afluencia en ciertos locales y al crecimiento en zonas urbanas gracias a la transformación digital.

En el contexto en el que se ha realizado este proyecto, la pandemia de COVID-19 ha causado en la sociedad actual problemas en muchos aspectos. Esto ha propiciado la búsqueda e implementación de soluciones para tratar de sortearlos, intentando reducir al máximo el impacto en el día a día. Sin embargo, ha sido inevitable tener que realizar cambios en aspectos tan básicos como la movilidad en aras de la seguridad.

De esta manera nació la necesidad de llevar un control de aforo. Aunque no es una idea nueva[1], hay que admitir que no estaba tan extendida como en la actualidad y ha tenido un gran auge en los últimos años. A pesar de esto, los modelos más usados suelen ser costosos, o consisten en un contador humano en los accesos al recinto contabilizando las personas que entran y salen.

Por todo esto, se planteó crear un sistema en tiempo real de control de personas en espacios cerrados que fuese asequible y automático. Estos dos conceptos principales fueron los que impulsaron la realización de este trabajo. Gracias a los avances tecnológicos se puede llegar a mejorar, abaratar y hacer llegar a todos los públicos este sistema.

[1] https://www.elindependiente.com/tendencias/2019/12/30/puerta-sol-blinda-campanadas-cuatro-filtros-policiales-maximo-19000-personas/

1.2. Objetivos

Se siguieron una serie de pautas para desarrollar el proyecto, y así, poder sacar una conclusión en base a ellas. Para cumplir con la tarea, se puede subdividir el objetivo principal en las siguientes premisas:

- El sistema ha de funcionar en tiempo real, lo cual implica que se está supervisando el entorno de manera continua y, al detectar un nuevo evento, este debe ser capturado y transmitido en el momento preciso en el que ocurre.
- El sistema debe tener el mayor rendimiento dentro de lo factible, ciñéndose a un bajo coste y aprovechando los componentes disponibles, evitando así, el uso de aceleradores gráficos como gasto adicional. Se pretende obtener la mayor rentabilidad posible y que esta, se ajuste al resto de objetivos sin necesidad de un coste extra.
- Para facilitar su uso, el sistema ha de tener una interfaz gráfica y ser multidispositivo. Esto implica que, los datos recogidos deben ser accesibles y de fácil visualización. Y así, por ejemplo, se pueden mostrar los datos de un local para sus administradores.
- Se debe conocer en todo momento la ocupación dentro de una zona cerrada. No solo eso, sino que además se puedan realizar mapas de calor de las distintas zonas del mismo, de manera que, se conozcan los lugares por los que hay un mayor aforo.
- Se debe garantizar la estabilidad del sistema, para que sus datos no dejen de ser accesibles, o algún componente se pueda ver afectado en el caso de que hubiera una ingente cantidad de datos a recoger y enviar.

1.3. Plan de Trabajo

Este proyecto comenzó a desarrollarse en el mes de julio de 2020. Como modelo, se siguió una metodología de trabajo ágil, ya que pareció conveniente tener reuniones entre los estudiantes cada poco tiempo. De esta manera, se otorgaba la oportunidad de poder establecer una hoja de ruta común para los plazos establecidos y compartir los avances e inquietudes que se tuvieran. De igual manera, cada mes se convocaba una reunión entre estudiantes y tutores, para comprobar el trabajo realizado y así poder verificar que dichos avances iban por el buen camino o si debían realizar algún cambio.

Para entender la cronología, se puede agrupar el trabajo desarrollado por los estudiantes en las siguientes etapas:

Primera etapa - Trabajo previo

Esta etapa está conformada por el tiempo transcurrido desde la primera reunión entre los estudiantes y los tutores, hasta mediados del mes de agosto. Durante esos dos meses, se estuvo realizando una investigación previa al propio inicio del proyecto, en la que se buscaron distintos sistemas de detección de personas. De esta forma, se podía encauzar el proyecto hacia la vía más conveniente, en este caso, la que cumpliera los objetivos descritos en el punto anterior.

Segunda etapa - Desarrollo

Esta etapa se centra en la implementación de las diferentes partes del modelo, en cuanto a los caminos tomados de manera simultánea y a las acciones realizadas.

El desarrollo de la aplicación web se enfocó en la creación de diferentes funcionalidades, de tal forma que se permitieran varios roles y todos los elementos de diseño tuvieran un carácter *Responsive Web Design*. Así, se podría adaptar la web a diferentes dispositivos y se lograría interactuar con esta en cualquier lugar.

La detección de Entrada-Salida (E/S) se enfocó hacia la mejora del código, para que pudiera ser ejecutado en una Raspberry Pi (RPI) o un dispositivo de características similares. La idea principal de esta parte era mejorar la utilidad de esta detección, sin perder eficiencia en los resultados.

Siguiendo el desarrollo, procede hablar de los sensores y dispositivos de comunicación, los cuales hubo que entender primero antes de poder adquirirlos. Se comenzó haciendo pruebas en una subetapa, con el sensor Sensor de Infrarrojos Pasivo (PIR) (mitad de etapa), la cual no sería continuada hasta casi terminar el modelo por completo.

De manera síncrona a las subetapas anteriores, se fueron desarrollando los módulos necesarios para establecer la comunicación. Se establecieron funciones dentro de cada programa, para poder comunicar los distintos lenguajes que fueron utilizados en el resto de módulos. Previo a esto, fue necesario realizar un estudio sobre las opciones disponibles, así como diferentes pruebas hasta llegar a ajustar adecuadamente el método de comunicación.

Tercera etapa - Documentación

La etapa de documentación se llevó a cabo en diversos puntos de la realización del proyecto. Para empezar a realizar el trabajo, los estudiantes tuvieron que prepararse adecuadamente, para ello, se procedió a documentar los hallazgos y las conclusiones a las que se llegaron.

De manera ocasional, se realizaba algún apunte para recordar cómo se habían hecho

ciertas instalaciones de programas informáticos o avances trascendentales para el proyecto. Sin embargo, no fue hasta el final de la etapa de desarrollo cuando se volvió a hacer énfasis en la memoria. De esta manera, se podían realizar mayores avances en el proyecto, fuera en el ámbito de desarrollo web, mapa de calor, detección E/S con vídeo, Internet of Things (IoT) o detección E/S con sensores.

Cuando solamente quedaban por realizar pequeñas correcciones sobre el proyecto, se continuó trabajando en la documentación. Mientras tanto, se realizaban una serie de pruebas sobre el modelo desarrollado por los estudiantes para determinar el correcto funcionamiento del mismo.

Cuarta etapa - Pruebas

A medida que se avanzaba en la parte de desarrollo, se realizaron pruebas para verificar el funcionamiento del modelo planteado y comprobar con qué ajustes sería más óptimo. Paulatinamente se fueron intensificando las pruebas para poder obtener una masificación de datos resultantes a comparar entre sí, de forma que se priorizaron unos avances sobre otros. Al finalizar el desarrollo total, se realizaron pruebas de resistencia para comprobar la estabilidad del sistema.

Es importante definirlo de forma más visual, por lo que se adjunta un diagrama de Gantt (ver Tabla 1.1 y Figura 1.1), en el que se muestran las fases anteriores y los hitos más destacables.

1.4. Organización de la memoria

Para simplificar la comprensión del trabajo realizado por el grupo, se ha escrito esta memoria de manera secuencial. Esta, está dividida en seis capítulos y cuatro apéndices.

En el primer capítulo, siendo esta la última sección, se explican las principales motivaciones y objetivos que han identificado este trabajo de fin de grado. Además del plan de trabajo seguido, asegurando la gestión de recursos disponibles de forma óptima.

En el segundo, se procederá a realizar un estudio de las tecnologías que se emplean en la actualidad. Con esto compararemos el distinto hardware en el que podría apoyarse la realización del trabajo de fin de grado (TFG) teniendo presentes los objetivos marcados. Asimismo, se expondrá algún ejemplo sobre el tema que se está tratando.

Posteriormente se continuará enumerando y explicando las tecnologías que han sido usadas para construir el modelo, y se dará una explicación de las mismas en el tercer capítulo. Concretamente, este se subdividirá según la categoría en la que han sido usadas. De esta

forma, quedará con cuatro grupos principales: web, IoT, sensores e inteligencia artificial (usada para la detección de la E/S de un recinto cerrado y los cálculos correspondientes para generar un mapa de calor).

El capítulo cuarto describirá la estructura del modelo, los escenarios en los que puede ser aplicado y su escalabilidad. También se analizarán el rendimiento y otros aspectos relevantes del mismo.

A continuación, en el capítulo cinco, serán definidos los límites del proyecto y se realizará una batería de pruebas que ayudará a documentarlos y justificarlos.

El apartado seis contendrá las conclusiones a las que se ha llegado tras el proyecto, se señalarán posibles mejoras o formas en las que sería posible avanzar y ampliar el trabajo a futuro con nuevas características o aumentar su funcionalidad.

Los capítulos séptimo y octavo serán el primero y el sexto traducidos al inglés, de manera que las conclusiones a las que se han llegado mediante la realización de este trabajo lleguen a un mayor público.

Por último, se podrán encontrar apéndices donde será ampliada información que nos parece relevante destacar o resaltar, como por ejemplo, el trabajo realizado por cada miembro.

Identificador	Hito
A	*Finalización trabajo previo*
B	*Desarrollo*
B1	Finalización desarrollo Python
B2	C++
B21	Primer código funcional
B22	Primera prueba completa en RPI
B23	Compilación TensorFlow-Lite API (TFL) en RPI
B24	Finalización mapa de calor
B25	Skip frames implementado
B3	IoT
B31	Búsqueda de información sobre Message Queuing Telemetry Transport (MQTT)
B32	Comparación de diferentes métodos
B33	Búsqueda de datos en diferentes lenguajes de programación
B34	Monitorización y configuración del broker
B35	Envío de mensajes mejorando la escalabilidad
B36	Envío de una matriz de datos para representar el mapa de calor
B37	Envío de una imagen sobre la cual pintar el mapa de calor
B40	Sensores
B41	Realizado modelo para sensor PIR
B42	Conexión de los sensores con la NodeMCU, y esta al broker
B43	Realizado modelo para sensor ultrasónico
B5	Web
B51	Crear Página principal
B52	Comunicación con Broker
B53	Actualización en tiempo real de aforo
B54	Crear Tablas BBDD, comunicación y persistencia
B55	Crear Página con gráficos
B56	Añadida seguridad para usuarios
B57	Recibir y mostrar mapa de calor
B58	Definir parametros Configurables y forma de editarlos
B59	Maquetación y diseño
B60	Tests y resolución de bugs
C	*Documentación*
C1	Documentación de investigación inicial previa al comienzo del proyecto
C2	Documentación de diversos avances
C3	Realización memoria
D	*Pruebas*
D1	Python
D2	C++
D3	IoT
D4	Sensores
D5	Web
D6	Conjuntas

Tabla 1.1: Hitos realizados del proyecto durante el proceso.

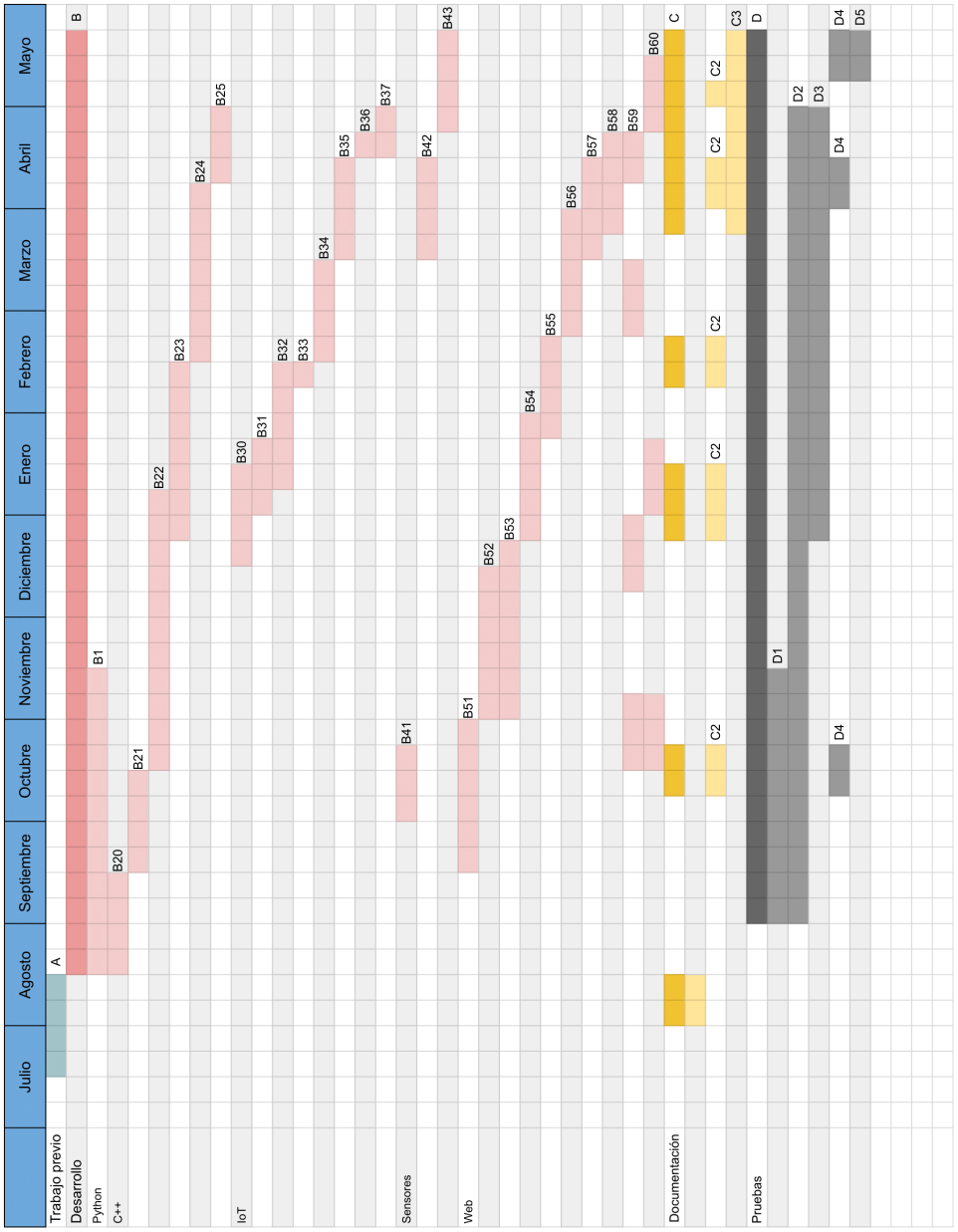


Figura 1.1: Diagrama de Gantt seguido en el desarrollo del trabajo.

Capítulo 2

Estado del Arte

2.1. Control de Aforos

Previa a la realización del proyecto, se procedió a hacer una investigación de campo sobre las diferentes herramientas disponibles relacionadas con el control de aforos y sus principales características. Una vez realizadas, se agruparon en varias categorías definidas por la tecnología empleada para captar o medir la ocupación. Estas son: sistemas con detección WiFi, detección ultrasónica, detección en estéreo, detección por infrarrojos y detección mediante cámara.

Sistemas con detección WiFi

Este modo de detección se basa en el registro de Media Access Control Identifier (MAC ID) de cada smartphone que accede o sale del local. Así entre otras cosas, otorga la posibilidad de saber cuantas personas se encuentran en el interior y al estar estas identificadas con un número único, cuánto tiempo dura su visita.

En este sistema, las MAC ID se obtienen mediante la captación de dispositivos con la señal WiFi activa (ver Figura 2.1). Por este motivo, resultan evidentes algunos de sus inconvenientes. Por ejemplo, la falta de un smartphone o la posesión de varios, no tener activada la conexión WiFi o cambiar la MAC ID[1] haría que el conteo no fuese preciso.

[1] https://blog.elevensoftware.com/how-mac-address-randomization-can-affect-the-wifi-experience

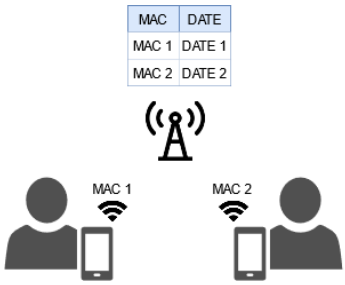


Figura 2.1: Concepto de detección por WiFi.

Sistemas con detección ultrasónica

La detección ultrasónica consiste en mapear una estancia basándose en la emisión y recepción de pulsos ultrasónicos a frecuencias que no pueden ser captadas por el oído humano (a partir de 20 KHz). Se puede mapear la distancia de cada ente basándose en el tiempo que transcurre entre emisión y recepción de las ondas. También influye la fuerza y el ángulo con las que son recibidas (ver Figura 2.2).

Para ello, estos sensores se sirven de esta propiedad y calculan el tiempo que tardan las ondas desde que son emitidas, hasta que las reciben de vuelta. Para generar dichas señales, los sensores hacen oscilar un cristal, basándose en un efecto piezoeléctrico de deformación que estos experimentan al introducirse en un campo eléctrico [5, pág 61].

Como contrapartida, por sí solo no sería capaz de distinguir entre personas y otros obstáculos. Además, ciertos materiales podrían atenuar o alterar las ondas de forma que no darían una lectura real.

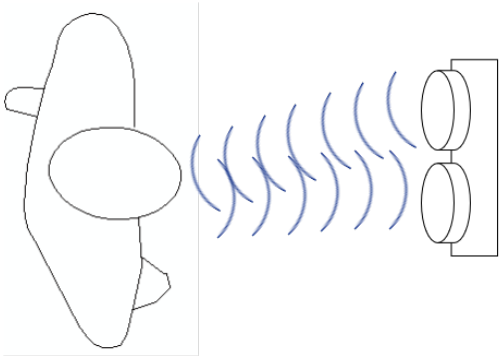


Figura 2.2: Concepto de detección por Ultrasonido.

Sistemas con detección por infrarrojos

Este sistema utiliza luz infrarroja para activarse y lanzar una señal cada vez que detecta el movimiento dentro de su campo de percepción (ver Figura 2.3). Los sensores más usados para esto son los PIR, que disponen de un sensor piro eléctrico (“Pyroelectric Infrared”)

capaz de captar radiación infrarroja y convertirla en una señal eléctrica. Son especialmente efectivos cuando el desplazamiento es perpendicular a su dirección de detección.

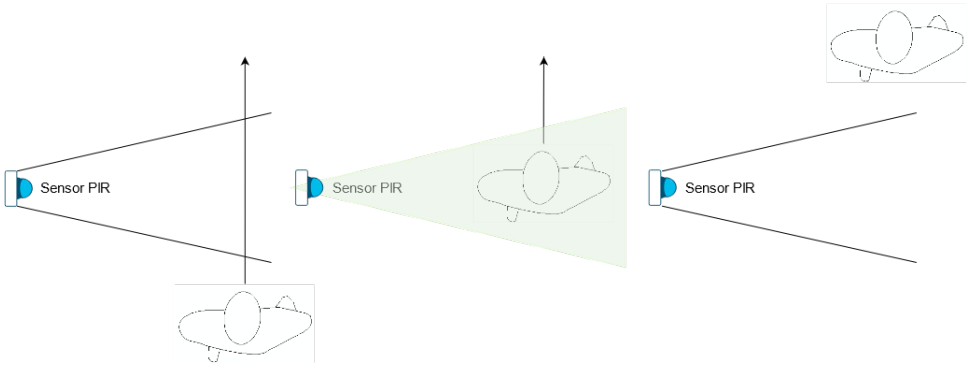


Figura 2.3: Concepto de detección por PIR.

La lente de Fresnel es un encapsulado con forma semiesférica fabricado con polietileno de alta densidad (ver Figura 2.4), cuyo objetivo es permitir el paso de la radiación infrarroja en un determinado rango (8 y 14 micrones). La lente detecta la radiación en un ángulo de 110º y de longitud de 3 a 7 metros, por lo que su campo de funcionamiento es limitado por donde se sitúe dicho sensor. Adicionalmente, la lente concentra la energía en la superficie de detección del sensor PIR, permitiendo una mayor sensibilidad del dispositivo [1, pág 2].

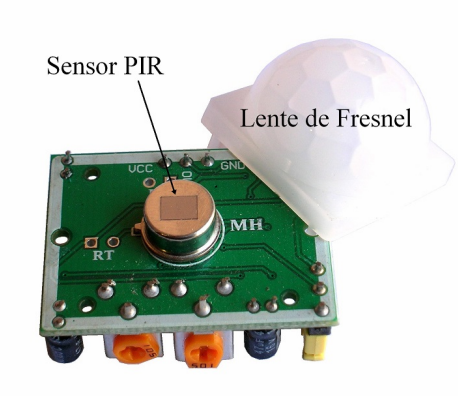


Figura 2.4: Esquema de elementos del sensor HC-SR501 [1].

Como aspectos positivos, tomando en cuenta los finalidades de este proyecto, se valorará que el uso de sensores PIR es barato, fácil de implementar y funciona en tiempo real. Sin embargo, pueden fallar, debido a que si alguien se queda en el campo de "visión" el suficiente tiempo, estos acabarían apagándose y dando un falso negativo.

Sistemas con detección óptica

Este método se suele compaginar con Inteligencia Artificial (IA) para simular la propia visión humana. De forma que, la cámara recoge la imagen y esta es procesada, ya sea para el conteo o seguimiento de individuos (ver Figura 2.5).

Identificar claramente a una persona, contabilizar entradas o salidas del recinto y seguir sus acciones, son algunas ventajas de estos sistemas. Además, se reducen problemas de solapamiento, como cuando varios individuos entran o salen al mismo tiempo de la estancia.

Una clara desventaja, es que el campo de visión de la cámara debe ser claro, sin tener elementos que dificulten la visión, y las redes neuronales deben estar lo mejor entrenadas posible.

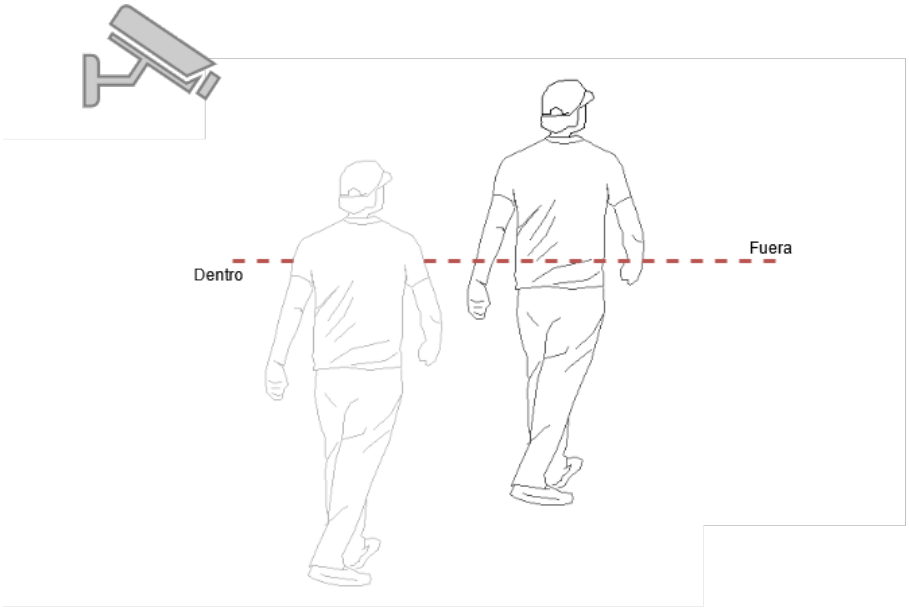


Figura 2.5: Concepto de detección por cámara.

Sistemas con detección en estéreo

El funcionamiento de este procedimiento, se centra en la utilización de dos receptores de imágenes equidistantes, tomando datos de manera simultánea. A consecuencia de esta medición, se pueden generar imágenes en 3D para poder hacer un seguimiento de un objeto [6].

La parte más cuestionable, no entrando en los objetivos definidos previamente, es el alto consumo de recursos al tratar dos imágenes de manera simultánea, haciendo una extracción de información de cada una. Este concepto se llama visión estereoscópica, con ella se puede hacer un control de aforo.

Comunicación de datos

Para transmitir toda la información captada por los sensores necesitan estar conectados. Un requisito del sistema es recopilar los datos de manera segura, para esto se requiere de unos módulos que interconecten con otros que pueda analizarlos y mantenerlos.

- Puede darse el caso de que los sensores y el módulo de tratamiento de los datos no se encuentren en la misma habitación; por lo que habría que conectarlos mediante cables, que pueden romperse si no se instalan de manera adecuada.
- Podría ser que los sensores y el módulo estén en la misma habitación, pero por algún motivo, entorpezcan el paso de las personas, o bien por problemas de espacio o por motivos de seguridad, no se pueda poner dicho módulo cerca.

NodeMCU

Para solventar los problemas expuestos anteriormente, se ha utilizado la placa NodeMCU, que es de código abierto tanto a nivel hardware como software. Su objetivo principal (a parte de la comunicación de datos vía WiFi de 2.4GHz), es la de facilitar la programación de los componentes que la forman. A continuación se exponen las distintas versiones y sus principales diferencias:

- **V1**: la primera generación (ver Figura 2.6)[2], la cual monta un chip ESP8266-12. Esta placa actualmente está obsoleta por lo que no se entrará en detalles, pero fue la precursora de las siguientes.

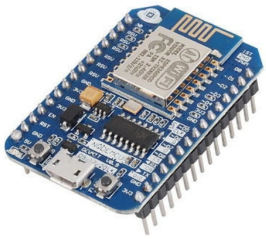


Figura 2.6: NodeMCU v1.

- **V2**: la segunda generación (ver Figura 2.7)[3] monta un chip ESP8266-12E. A pesar de disponer de más pines, ocupa menos espacio que la placa anterior. Esta placa de desarrollo no podrá ser utilizada para este proyecto pues no soporta alimentación de 5V, que es lo necesario para que funcione el sensor PIR.

[2] https://descubrearduino.com/nodemcu/
[3] https://electronics.semaf.at/NodeMCU-v2-ESP8266

Figura 2.7: NodeMCU v2.

- **V3**: la llamada "tercera generación" (ver Figura 2.8)[4] en realidad no lo es, ya que es una versión realizada por distintos fabricantes. Integran el chip ESP2866, aunque la versión de la empresa "Lolin" utiliza el *12F*, y el de la empresa "Lua" el *12E*. Esta sí que incorpora un pin para alimentación a 5V, aunque es más grande que la anterior.

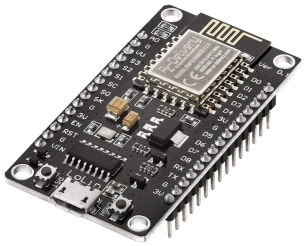


Figura 2.8: NodeMCU v3.

Chip ESP32

Para terminar, no se debe acabar este apartado sin realizar una serie de apuntes sobre el chip *ESP32*, ya que, en función de una serie de parámetros, puede ser mejor emplear este chip que el *ESP8266*. Si bien ambos tienen WiFi y un procesador de *32* bits, en la Tabla 2.1[7] se comparan sus características más destacables (y de utilidad en este proyecto) .

Característica	ESP32	ESP8266
CPU	Dos núcleos entre 80-240MHz	Un núcleo 160MHz
WiFi	Hasta 150Mbps	Hasta 72'2Mbps
GPIO	36 pines	17 pines
Bluetooth	Sí	No
SRAM	520kB	<50kB
ROM	448kB	No

Tabla 2.1: Comparativa entre los chips ESP32 y ESP8266.

[4] https://ssdielect.com/cb/wifi-sistemas-de-desarrollo-esp8266-wifi/768-nodemcu-lolin-ch340.html

Para este proyecto se utilizarán los sistemas de detección óptica y ultrasonidos, por ser los que más concuerdan con los objetivos definidos en relación al bajo coste, a la detección en tiempo real, a la generación de un mapa de calor y a la escalabilidad de los dispositivos. Además, para la comunicación de datos será utilizada la placa NodeMCU v3, la cual posee el chip *ESP8266-12F*.

2.2. Visión Artificial

Este campo de la IA es uno de los más llamativos del último lustro. Esto se debe a la gran cantidad de aplicaciones que tiene, como pueden ser la automoción, el control de aforos, o incluso saber el estado emocional de una persona con sólo ver su expresión facial.

A pesar de que la visión artificial tiene un campo de investigación amplio y se aplica a nuevas tecnologías, es aconsejable acompañarlo de otros sensores y dispositivos, y así, aumentar la eficiencia de los resultados obtenidos.

Un ejemplo de aplicación concreto, que hace referencia a la olivicultura [8], consiste en un estudio mediante Visión por Computador (CV) de campos sobrevolados por drones con cámara integrada. Estos detectan el estado de la oliva dentro de cabinas y realizan un conteo de árboles en una determinada zona. Todo esto se llevó a cabo con herramientas como MATLAB Image Processing Toolbox y MATLAB Deep Learning Toolbox. También, se ha llegado a desarrollar software, donde mediante CV, se consigue que un robot pueda llegar a resolver un cubo de Rubik [9].

La ocupación en zonas cerradas se puede usar para estudios relevantes. En particular, se ha llegado a controlar la refrigeración dentro de un edificio con ella [10]. En las pruebas realizadas en ese proyecto, gracias a CV, se pudo controlar el aforo y además reducir hasta en un *59 %* la energía consumida.

Para poder obtener información relevante a partir de una imagen y Redes Neuronales (RRNN), son necesarios sistemas o una serie de librerías que ejerzan dicha función. A continuación, se presentarán una serie de herramientas utilizadas para CV (ver Tabla 2.2).

Una vez estas sean expuestas, se tendrán en cuenta los ejemplos que guarden relación con el tema de este proyecto y se aplicarán las herramientas que sean de código abierto para cumplir el objetivo de conseguir el menor coste posible. En este caso, las seleccionadas son TFL y Open Source Computer Vision (OpenCV).

Logo	Características
	▪ TensorFlow (TF) es una librería desarrollada por Google, que toma como objeto los tensor cores para realizar operaciones de detección y entrenamiento de redes neuronales.
	▪ OpenCV es una de las librerías más usadas junto con TensorFlow debido a que ambas son de código libre. Esta herramienta es muy potente pero aún sigue en desarrollo. Tiene expectativas esperanzadoras y es muy fácil de usar. Es sencillo compilarlo en diferentes dispositivos.
	▪ Amazon Rekognition es una aplicación de pago ofrecida por Amazon Web Services. Este sistema cuenta con una gran herramienta de detección de imagen, aunque es algo costosa y aún tiene desarrollos pendientes, según la opinión de muchos de sus clientes.
	▪ Microsoft Azure Computer Vision cuenta con funcionalidades fáciles de usar, aunque carece de funcionalidad debido a su estado aún en desarrollo. Según cuentan muchos usuarios, el reconocimiento de caracteres es algo deficiente.
	▪ IBM Watson Visual Recognition se trata de una herramienta desarrollada por IBM, que cuenta con grandes prestaciones, pero que pierde eficiencia a la hora de utilizarlo de manera local y no es fácil de implementar.

Tabla 2.2: Herramientas más usadas en CV.

2.3. Intercambio de Mensajes

A la hora de comunicar los datos obtenidos, gracias a la detección y conteo de personas con la aplicación web, se barajaron diferentes opciones. Debido a que podría haber distintos dispositivos enviando sus datos simultáneamente, se llegó a la conclusión de que usar el protocolo Message Queuing Telemetry Transport (MQTT) era lo más adecuado. Para utilizar este protocolo es necesario un broker que reparta los mensajes, los más populares son los que se muestran en la Tabla 2.3.

Logo	Características
	▪ Mosca es un broker para Node.js, ligero y desarrollado en javascript.
	▪ ActiveMQ es un broker JMS (Java Message Script) desarrollado por Apache.
	▪ ZMQ es una librería de código abierto utilizada para la interconexión de aplicaciones. Con esta librería se puede desarrollar también este protocolo sin la necesidad de un broker intermedio.
	▪ Mosquitto es un broker ligero de código abierto que implementa MQTT en sus versiones *5.0* y *3.1.1*. Se utiliza en todos los dispositivos con hardware de bajo coste. Ofrece una gran variedad de librerías para su desarrollo pero las principales son: • **Libmosquitto:** librería para desarrollar la conexión con el broker en C y C++. • **Paho:** librería para desarrollar la conexión con el broker en Java y Python.

Tabla 2.3: Brokers populares para MQTT.

Tras estudiar las diferentes librerías que aplican el protocolo MQTT, se ha escogido la librería de Mosquitto debido a tener una gran versatilidad entre diferentes lenguajes y por poseer un broker intermedio (a diferencia de ZMQ).

2.4. Hardware de bajo consumo y coste

Actualmente, la posibilidad de detectar a personas en tiempo real supone un coste económico y computacional bastante elevado. Es por eso que uno de los objetivos principales de este proyecto era obtener el máximo rendimiento con el menor coste posible.

Como se ha explicado con anterioridad, existen varias formas para realizar el conteo de personas y cada una de ellas emplea una gran variedad de dispositivos. Estos dispositivos son por ejemplo los que se mencionan en Tabla 2.4.

Dispositivo	Descripción	Precio (€)
Raspberry Pi 4	Miniordenador que cuenta con un procesador de cuatro núcleos y hasta 8GB de *RAM*	56,99 - 91,90
Banana Pi BPI-M4	Miniordenador que cuenta con un procesador de cuatro núcleos, con 8Gb de memoria eMMC y hasta 2GB de RAM	48,78

Tabla 2.4: Microcontroladores más comunes.

Para la detección a través de sensores, los controladores más utilizados son los reflejados en la Tabla 2.5.

Dispositivo	Descripción	Precio (€)
Arduino Nano	Microcontrolador ATmega328, es de un tamaño reducido y ligero	20,00
Arduino Mega	Microcontrolador ATmega2560 y con 56 pines de I/O	35,00
Arduino Uno	Microcontrolador ATmega328P y con 14 pines de I/O	20,00
Node MCU V2	Placa Arduino con microprocesador ESP8266 y chip WiFi ESP12E	9,80
Node MCU V3	Placa Arduino con microprocesador ESP8266 y chip WiFi ESP12F	7,70

Tabla 2.5: Controladores más utilizados.

Para el conteo de personas se utilizan principalmente los sensores que se muestran en la Tabla 2.6.

Dispositivo	Descripción	Precio (€)
HC-SR501	Sensor PIR	2,90
HC-SR04	Sensor de Ultrasonido	5,77

Tabla 2.6: Sensores más comunes.

Para la implementación de este tipo de proyectos, en la actualidad, se emplean aceleradores gráficos como el Coral USB o el Jetson Nano. Pero al incluirlo en este proyecto, se dejaría a un lado la idea principal de obtener el máximo rendimiento con el menor coste (ver Tabla 2.7).

Dispositivo	Descripción	Precio (€)
Coral USB	Acelerador USB muy utilizado junto con los dispositivos de bajo coste ya mencionados para incrementar la potencia y el procesado de imágenes	49,32
Jetson Nano	Kit de desarrollo creado para IA	88,90

Tabla 2.7: Aceleradores más utilizados.

Teniendo en cuenta la serie de herramientas de hardware explicadas, se van a seleccionar una serie de componentes los cuales podrán ser aplicados al proyecto. Habrá que tener en cuenta qué coste y qué tipo de hardware ofrecen empresas referentes a este tema. El coste ofrecido actualmente, se estima entorno a unos *1500*€, hasta precios cercanos a los *3000*€[5], donde se satisfacen necesidades de control de aforo E/S con cámaras estereoscópicas.

Por esta razón, se propondrá una tabla de precios asociada a los dispositivos seleccionados, como se puede ver en la Tabla 2.8 y así, obtener una suma total de coste inferior al *10* % sobre el precio de mercado.

Dispositivo	Precio (€)/Unidad	Unidades	Precio (€)
Raspberry Pi 4 - 4 GB	59,90	1	59,90
Node MCU V3	7,70	2	15,40
HC SR-501	2,90	2	5,80
HC-SR04	5,77	1	5,77
		Total	86,87

Tabla 2.8: Importe y cantidad de los dispositivos seleccionados.

[5] https://www.mercasat.es/1569-control-de-aforo
https://www.planafabrega.com/camara-ip-minidomo-4mp-optica-vfm-28-12mm-ir-30m-conteo-personas-reconocimiento-facial
https://jmseguridad.es/control-de-aforo/

Capítulo 3

Arquitectura y tecnología empleada

3.1. Desarrollo Web

Previamente se ha explicado, que una de las motivaciones de este proyecto es la de ser usada por alguien que no tenga un alto nivel de conocimiento informático. Así, surgió la necesidad de crear una interfaz para abstraer lo máximo posible la funcionalidad.

Para conseguir este objetivo, se ha implementado una aplicación web con dos características principales: ser multiplataforma y permitir a un usuario con más privilegios, el acceso a estadísticas del aforo registrado o la configuración de ciertos parámetros, que se explicarán más adelante.

3.1.1. Tecnologías Web

Lenguajes

- Java

  El principal lenguaje usado en el desarrollo de esta aplicación es Java, ya que es un lenguaje multiplataforma orientado a objetos. Ambas cualidades concuerdan con las necesidades previamente expuestas para este proyecto.
  Además, al tener que utilizar una base de datos, cuya estructura y características serán explicadas en próximas secciones. El uso de este lenguaje ha permitido utilizar un framework como Hibernate, el cual facilita la persistencia y recuperación de datos.

- HTML

  Este es otro de los lenguajes empleados a la hora de desarrollar el código de la aplicación web. Sin embargo, cuantitativamente, debido al uso de Apache Wicket,

la cantidad de código desarrollado es mucho menor que la de Java por la forma en la que este funciona. El código HTML solo contiene las etiquetas e identificadores necesarios para poder ligarlo al propio código Java.

Frameworks

Se han empleado principalmente dos frameworks para facilitar y agilizar el desarrollo de la aplicación, estos serán explicados a continuación.

- Apache Wicket

  Wicket es un framework orientado a componentes. Es decir, permite encapsular el código para que sea más fácil su reutilización y mantenimiento. Funciona coordinando Java y HTML de forma que, debe haber un fichero HTML con el mismo nombre y en el mismo directorio que el código Java.

  Cuando la aplicación está desplegada en el servidor y el usuario solicita una página, el servidor compila el modelo y posteriormente se genera el HTML basado en él [11]. Una vez generado, se le devuelve el resultado al usuario (ver Figura 3.1).

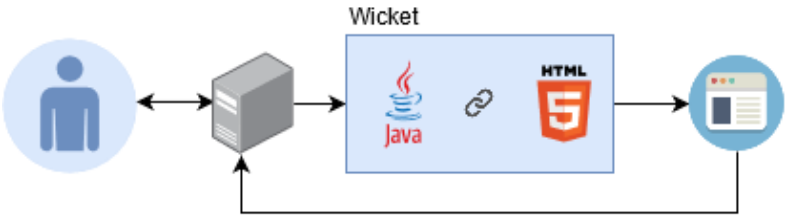


Figura 3.1: Esquema sobre el funcionamiento de Wicket.

  Otro aspecto positivo de Wicket, es el de facilitar el uso de Asynchronous JavaScript and XML (AJAX). Esto es útil porque hace que la aplicación pueda funcionar de forma asíncrona y procese las peticiones al servidor en segundo plano. También recarga el HTML o permite programar el comportamiento por eventos en una página sin necesidad de desarrollar JavaScript.

- Hibernate

  Para gestionar la conectividad entre la base de datos y la aplicación web se ha empleado el framework Hibernate, el cual es de tipo Object/Relational Mapping (ORM).

  ORM es un modelo de mapeo automático que permite asociar, de forma bidireccional, los objetos de una aplicación Java y una base de datos relacional. Para su funcionamiento, se ayuda de anotaciones tales como *@Entity*, *@Table* e *@Id* en las

clases del código Java y sus atributos, para ligarlas con las entidades, tablas y columnas de la base de datos mediante el uso de metadatos (ver Figura 3.2). Para el uso de esta tecnología se ha seguido el curso de persistencia de Baeldung [1].

Figura 3.2: Esquema sobre el funcionamiento de Hibernate.

Compilador

- Maven

  Es un compilador para proyectos basados en Java. Usa un fichero de extensión Extensible Markup Language (XML) (pom.xml) para gestionar las distintas dependencias del proyecto. Se encarga de validarlas, descargarlas y compilarlas en archivos de tipo Java Archive (JAR). Una vez las tiene, procede a hacer lo mismo con el proyecto y generar el fichero de tipo Web Application Archive (WAR), que servirá para desplegar la aplicación. Es una herramienta bastante útil que permite la gestión de dependencias de forma rápida e intuitiva.

Despliegue

- Tomcat

  Al tratarse de un proyecto Maven, tras compilar el código se genera un WAR que se desplegará utilizando un servidor Tomcat. Se ha utilizado esta tecnología porque es uno de los servidores y contenedores de servlets más conocidos y extendidos para Java. Esto implica que la aplicación web será desplegada y generada de forma dinámica en el lado del servidor [12].

3.1.2. Persistencia de datos

Todos los datos que son enviados a la página web han de ser almacenados para posteriormente poder consultarse, por lo que se ha de utilizar una base de datos para cumplir el cometido.

La base de datos está codificada en Structured Query Language (SQL), ya que proporciona mayor seguridad que otros lenguajes. Tiene una gran escalabilidad y estabilidad.

[1] https://www.baeldung.com/java-bootstrap-jpa

Todo esto se traduce en que la solución desarrollada pueda ser empleada en un mayor número de sitios, pudiendo integrarse con otras herramientas ya existentes.

El hecho de necesitar tablas dependientes entre sí, hizo que al final se utilizara una base de datos relacional en contraposición con las no relacionales.

MySQL es un sistema de gestión de bases de datos, y los motivos por los que se ha escogido son, entre otros (ver [13, pág 7]):

- Está optimizado para equipos de múltiples procesadores.
- Tiene una gran velocidad de respuesta.
- Se puede utilizar como cliente-servidor.
- Soporta múltiples métodos de almacenamiento de tablas, con distintas prestaciones y rendimientos para poder optimizarlo a gusto en cada caso concreto.
- Se tiene constancia de casos en los que se maneja cincuenta millones de registros y sesenta mil tablas y cinco millones de columnas, por lo que en cuanto a escalabilidad y crecimiento sus características encajan con lo buscado para este proyecto.
- Se puede conectar mediante protocolo TCP/IP, e incluso sockets (UNIX y NT), por lo que a futuro pueden desarrollarse soluciones con ellos.
- Es altamente confiable en cuanto a estabilidad se refiere.

Por todo esto, en vistas a mejoras y en relación al crecimiento de la base de datos, se ha pensado que lo mejor es utilizar MySQL.

3.2. Internet de las cosas

El concepto de IoT nace en 1999, de la mano de Kevin Asthon, tras realizar investigaciones en el campo de la identificación por radiofrecuencia en red y tecnología de sensores. Hace referencia a la posibilidad de conectar un conjunto de sensores y actuadores a una misma red, permitiendo el intercambio de mensajes entre los diferentes dispositivos. Para permitirlo, se ha empleado el protocolo MQTT que utiliza el patrón publicador-suscriptor. Este patrón hace uso de varios elementos, los cuáles se van a definir a continuación:

Cliente

Los clientes son todos los dispositivos conectados entre sí que enviarán o recibirán los diferentes mensajes de otros clientes. Comúnmente se les llama publicadores y suscriptores según si su tarea es enviar datos o recibirlos.

Broker

Un broker es un gestor que actúa como intermediario entre publicadores y suscriptores, recibiendo y enviando mensajes a los clientes. Es el encargado de administrar los temas en los que los "publishers" están publicando y a los que los "suscribers" están suscritos.

3.2.1. Patrón Publicador - Suscriptor

Antes de detallar en qué consiste el protocolo MQTT, se debe indicar cuál es la idea principal del patrón publicador-suscriptor. Con este, a diferencia de otros, no se utiliza la tradicional forma de comunicación cliente-servidor, sino que se hace uso de una comunicación máquina a máquina (M2M) mediante publicaciones y suscripciones a diferentes temas.

En estos temas será donde los publicadores envíen sus mensajes, estos llegarán al broker y se encargará de enviarlos a los respectivos suscriptores de cada tema, completando así la comunicación.

Por lo que para poder tener una comunicación se tiene que cumplir el siguiente esquema (ver Figura 3.3):

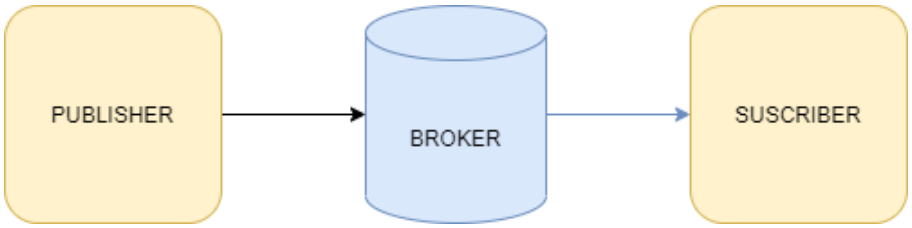


Figura 3.3: Ejemplo Publicador - Suscriptor.

3.2.2. Message Queuing Telemetry Transport

MQTT es un protocolo de código abierto para intercambio de mensajes muy sencillo, que hace uso de poco ancho de banda, por eso es muy utilizado en las aplicaciones de IoT. Este protocolo cuenta con una gran variedad de librerías para diferentes lenguajes, por lo que, facilita mucho la conexión entre múltiples aplicaciones.

Formato de los paquetes

Los paquetes de MQTT se componen de la siguiente manera (ver Figura 3.4):

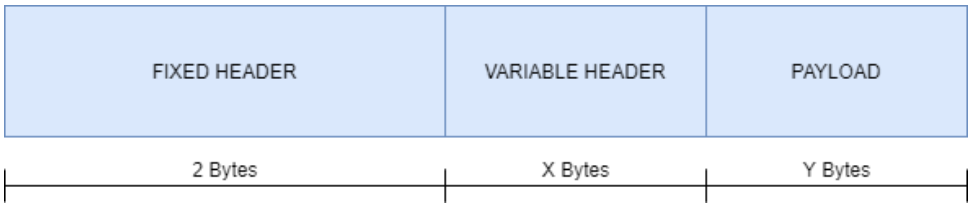


Figura 3.4: Formato Paquetes MQTT.

- Fixed Header: donde se determina el tipo de mensaje que se está enviando y su longitud. Este campo ocupa dos bytes.
- Variable Header: este campo varía según el tipo de mensaje, por lo que su contenido y longitud son variables.
- Payload: este campo también es variable, en caso de publicar un mensaje contiene el tema en el que se publica y el propio mensaje.

Calidad de Servicio

El tipo de comunicación establecida por este protocolo varía en función de la calidad de servicio (Quality of Service) que se elija. Este protocolo permite definir tres tipos de Quality of Service (QoS) las cuales se detallan a continuación.

- Solo una vez - QoS 0

  Si se define la QoS del sistema como 0, el mensaje se enviará una sola vez sin esperar confirmación del destinatario (ver Figura 3.5).

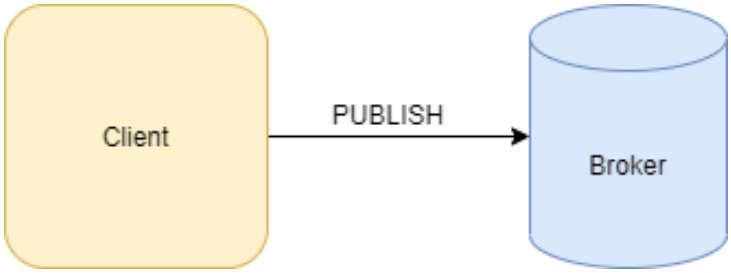


Figura 3.5: QoS 0: Solo una vez.

- Al menos una vez - QoS 1

  Este tipo de calidad de servicio se asegura de que el mensaje llega al receptor al menos una vez. En esta ocasión, el transmisor almacena el mensaje hasta que recibe un ACK de confirmación. En caso de no recibirlo, lo envía las veces que sea necesario (ver Figura 3.6).

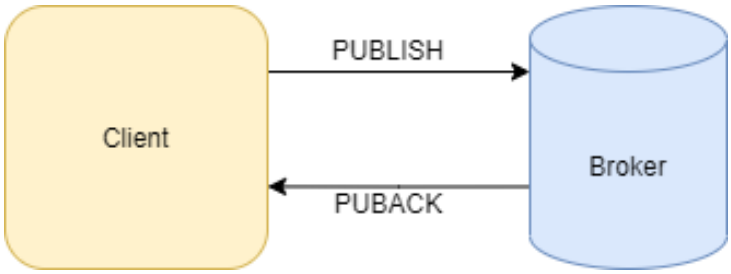


Figura 3.6: QoS 1: Al menos una vez.

- Exactamente una vez - QoS 2

  Es el mayor nivel de calidad de servicio que se puede seleccionar en este protocolo. Al establecer este nivel, el tipo de comunicación es el más seguro y el más lento, ya que garantiza que el mensaje llegue. El transmisor envía el paquete, el receptor responde con un *Publish Received*, si no llega al transmisor, este vuelve a enviar el mensaje, si lo recibe, envía un *Publish Realease* y el receptor responde con un *Publish Complete* (ver Figura 3.7).

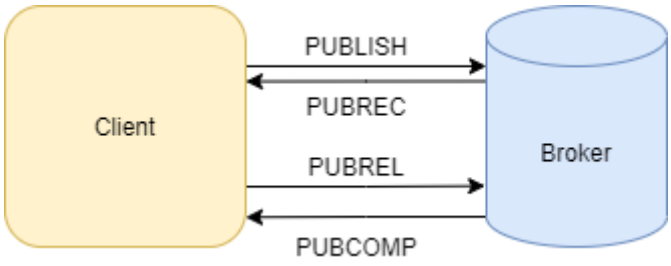


Figura 3.7: QoS 2: Exactamente una vez.

Filtros por tema

Como se ha indicado anteriormente, los mensajes en MQTT se envían a través de temas, también llamados topics, en los que se publican los mensajes. Los topics son cadenas de texto UTF-8 y de una longitud máxima de 65536 caracteres entre los que se distinguen mayúsculas y minúsculas. Estos topics están formados por varios niveles que se dividen entre sí separados por '/'. Para referirse a estos niveles, se puede usar el símbolo '+' si se pretende sustituir un único nivel, o el símbolo '#' si se busca sustituir varios.

3.2.3. Infraestructura

Este protocolo habitualmente hace uso de una topología de estrella ya que se compone de los diferentes clientes que actúan como publicadores. Estos se comunican con una máquina que actúa como broker repartiendo los diferentes mensajes a los suscriptores de los temas adecuados (ver Figura 3.8).

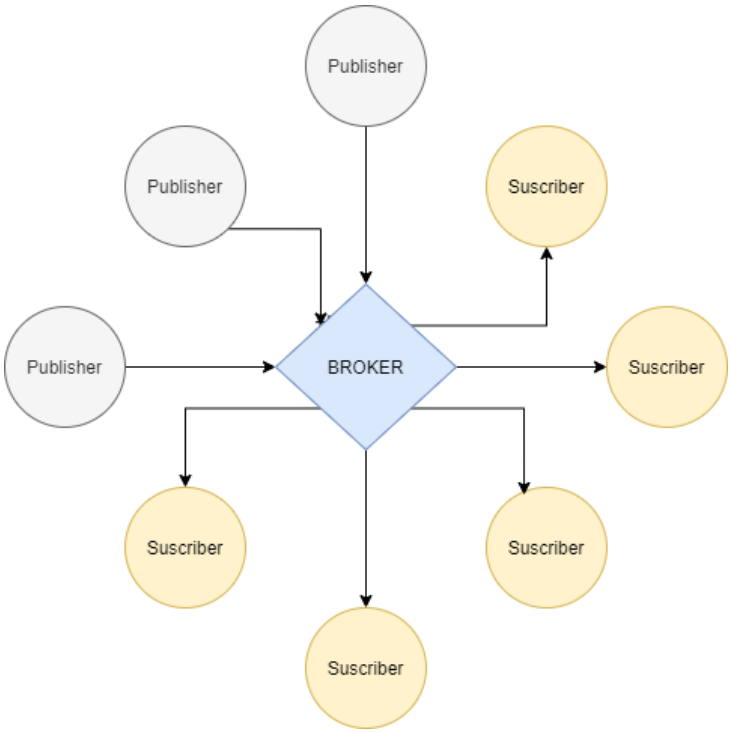


Figura 3.8: Modelo topología de estrella.

3.3. Reconocimiento y clasificación de imágenes

La Visión por Computador (también llamada *Visión Artificial*), perteneciente a una subdivisión de la IA, está asociada al tratamiento de imagen imitando la visión humana en cuanto a, no solo la percepción de colores y formas, sino también a su interpretación. Cabe aclarar que, para poder interpretar dichas imágenes, se necesitan tramitar ingentes cantidades de datos matriciales en 2 o 3 dimensiones.

Para poder identificar, clasificar y analizar objetos dentro de cada fotograma, se utilizan las RRNN. Una vez reconocidos los elementos, se tratan como cajas delimitadas por el espacio que ocupan. El principal objetivo es conseguir el mayor rendimiento de detección para poder hacer un seguimiento de las mismas. En estos aspectos se centrarán los siguientes puntos.

3.3.1. Redes Neuronales

Una red neuronal en el campo de la IA, es similar a una en el de la neurociencia [14]. De manera análoga, la neurona en IA, se asemeja en estructura y comportamiento a la neurona biológica. En ambos casos, la neurona posee una serie de entradas que son procesadas en el núcleo o soma [15] y dan lugar a una salida (ver Figura 3.9) que se conectará a la siguiente neurona.

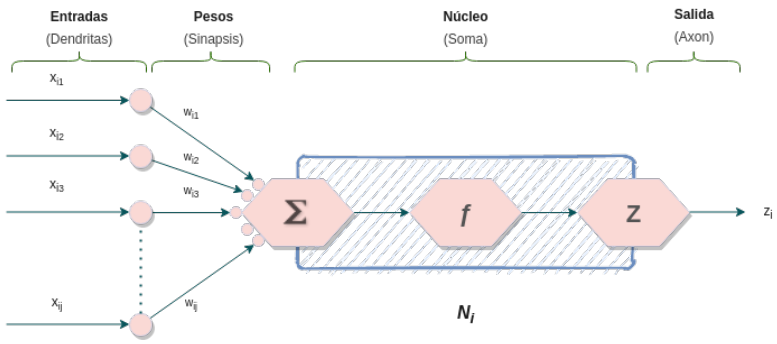


Figura 3.9: Partes de una neurona en IA.

Cada núcleo contenido en una neurona toma un proceso en el que intervienen: una función de entrada, dada por la expresión $n_{ij} * w_{ij}$, donde n representa las entradas, w los pesos, $*$ indica la operación de entrada, j el máximo de elementos e i indica a qué neurona se refiere; una de activación, asociada al tipo de red neuronal, y una de salida, que será el resultado final o servirá como entrada a otra neurona dentro de la red.

Una vez se ha visto el comportamiento de una neurona, se puede apreciar cómo actúan a modo de red neuronal [15, pág. 16], conectándose según los pesos de cada una. En la Figura 3.10 se define una capa de entrada, una lógica aplicada a una serie de capas ocultas y por último una capa de salida destinada a dar unos resultados de todo el proceso.

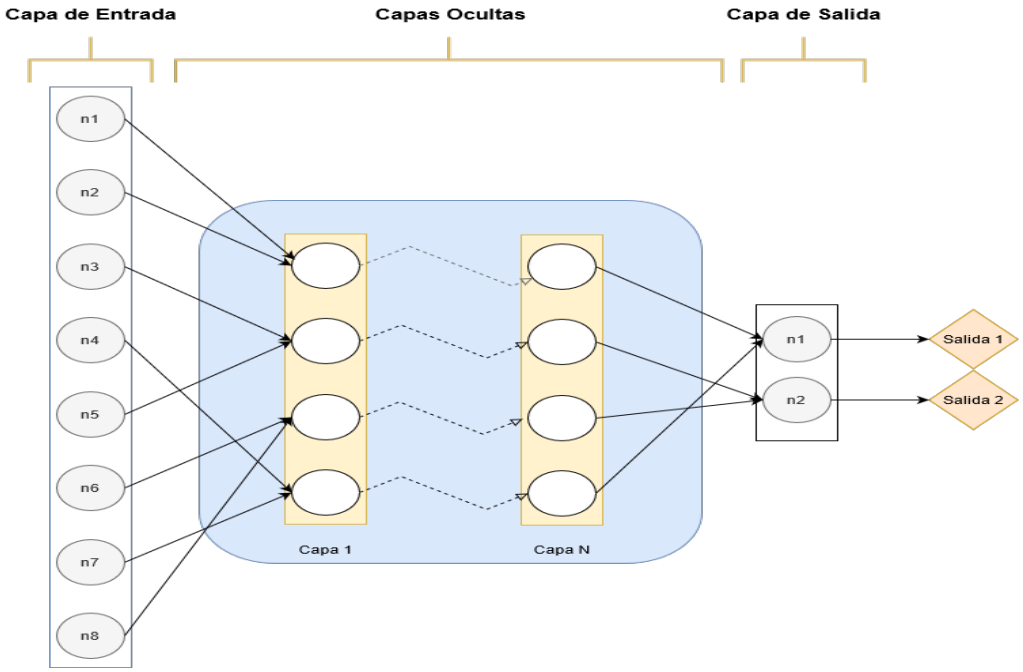


Figura 3.10: Estructura de una red neuronal.

Redes Neuronales en CV

Teniendo en cuenta el concepto básico, se van a explicar las redes neuronales utilizadas en este proyecto. Se detallarán las Faster R-CNN (FRCNN) [4] y otras empleadas en algoritmos tales como You Only Look Once (YOLO) o Single Shot Detector (SSD), siendo estas las arquitecturas más comunes en visión por computador.

Las Convolutional Neural Networks (CNN), Figura 3.11, son aquellas redes encargadas de tramitar una imagen y clasificarla, normalmente según si se reconocen en ella personas, animales u objetos. Por otro lado, se encuentran las Regional Proposal Networks (RPN), Figura 3.12, que definen los posibles objetos dentro de una imagen, haciendo una serie de proposiciones con un coste de ejecución ínfimo.

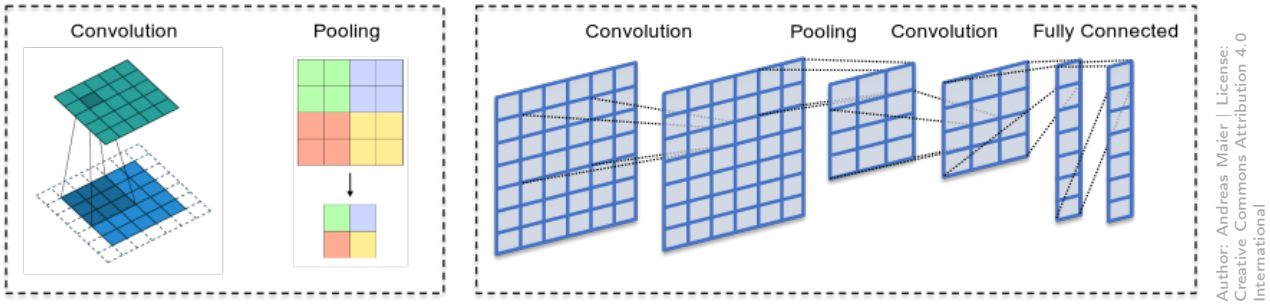


Figura 3.11: Representación del funcionamiento de una CNN.

Figura 3.12: Simulación del funcionamiento de una RPN, en cuanto a las coordenadas de una serie de proposiciones.

Una Faster R-CNN está compuesta por el uso de RPN junto a una Fast R-CNN para obtener el menor coste de ejecución. Primero se generan una serie de regiones de manera convolucional, que posteriormente atravesarán la parte de clasificación de objetos. Cabe destacar que, es más eficiente que sus competidoras R-CNN y Fast R-CNN (ver Tabla 3.1)[2].

	R-CNN	Fast R-CNN	Faster R-CNN
Tiempo de cada imagen(s)	50	2	0,2
Mejora	1x	25x	250x

Tabla 3.1: Comparativa R-CNNs [4].

Estas RRNN representan una arquitectura de dos fases, mientras que para la detección en tiempo real se emplean arquitecturas de una fase [16] como YOLO y SSD.

Visualizando la Figura 3.13, se entiende como arquitectura de una fase aquella que es capaz de realizar toda su funcionalidad traspasándola una única vez. Gracias a esto, en CV, se obtienen mejores resultados en cuanto al tiempo de detección, perdiendo por otra parte eficiencia en cuanto a clasificación sobre los resultados obtenidos.

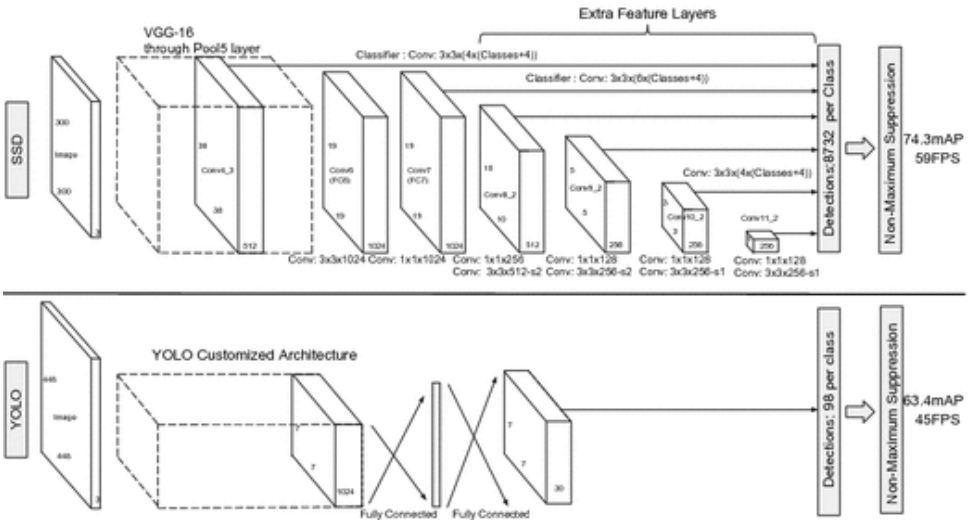


Figura 3.13: Redes neuronales con arquitectura de una fase, *SSD* y *YOLO* [2].

Para una ejecución de código sobre dispositivos de bajo consumo, se precisa obtener el mayor rendimiento posible, por lo que se priorizarán arquitecturas ligeras, como las ofrecidas partiendo de la base de YOLO. Tomando un estudio previo realizado donde se ha utilizado *TinyYOLOV3*, se tiene la siguiente conclusión sobre su uso sobre RPI [17, pág 71]:

[2] https://www.youtube.com/watch?v=v5bFVbQvFRk

Inferencia en imágenes y conjuntos de imágenes (si el tiempo total / FPS no es relevante) en los que puedan utilizarse redes ya entrenadas.

De este modo, no podríamos aplicar esta red neuronal sobre nuestro proyecto. Para poder emplear *TinyYOLOV3*, tendría que descartarse de manera automática el objetivo que trata sobre el tiempo real.

Considerando la Tabla 3.2, la eficiencia del algoritmo SSD destaca frente a la competencia empleando el mismo *dataset*[3]. La serie de pruebas realizadas sobre este tipo de RRNN son estudiadas en base a procesadores de consumo medio o elevado, por lo que se ha pensado aplicar una red neuronal más ligera basada en SSD.

	YOLO	SSD	Faster R-CNN
Fotogramas por segundo	45	59	7
mean Average Precision (VOC 2007)	63,4 %	74,3 %	73,2 %

Tabla 3.2: Comparativa entre las RRNN mostradas [2].

Más concretamente, la que se ha llegado a utilizar para este proyecto es ***SSD Lite Mobilenet V3*** partiendo del algoritmo SSD, en ambos casos se emplea la misma estructura, a diferencia del uso de pesos asociados a las neuronas. Mientras que SSD Lite utiliza pesos de enteros de 8 bits, SSD usa punto flotante de 12 bits. Con datos obtenidos a través de un estudio realizado [18] y lo expuesto anteriormente, se ha partido de una *SSD Mobilenet* hacia una tipo SSD Lite, y esto es, debido a su gran rendimiento de FPS[4] y una eficacia de detección, suficiente para el procesamiento de vídeo.

3.3.2. Frameworks

En esta sección se detallarán dos de las librerías que han sido aplicadas para poder hacer la detección de los objetos. Además, se harán comparaciones entre ambas, en función de los resultados generados en la RPI donde se va a ejecutar la detección.

OpenCV

La librería de OpenCV administra el tratamiento de imágenes para un mejor manejo de estas en el código. Incluye abundantes algoritmos, entre los cuales, hay una sección de backend, que ayuda a la detección con RRNN variadas. El concepto que tiene esta librería es el de integrar métodos de otros procesadores de imágenes para diferentes tipos de dispositivos. Cuenta con muchas opciones para realizar sus operaciones junto con

[3] Anglicismo referente a un conjunto de datos habitualmente tabulados.
[4] Fotogramas por segundo.

herramientas como OpenVino, sentencias OpenCL, o bien sentencias alojadas en una FPGA.

Además de esto, se pueden utilizar sus funciones para detectar objetos en imágenes que hacen muy cómodo su manejo y visualización. El problema de esta librería es su estado, aún en desarrollo, y algunas de sus funcionalidades están en fase alpha. Aún está en proceso de integración de código y algunos componentes todavía no son del todo eficaces para poder utilizarlo sin ayuda de otra librería.

TensorFlow

Librería de código abierto desarrollada por Google, al igual que OpenCV, es utilizada para detección de objetos y además, para entrenamiento de redes neuronales. El segmento que compete al entrenamiento de redes neuronales es utilizado junto con la herramienta de Keras. En el caso de este proyecto, se empleará la parte de detección de objetos que usa *tensor cores*, que no es más que un objeto matemático relacionado con otros (en este caso matrices multidimensionales).

3.3.3. Bounding Boxes

En este apartado se verán las funciones utilizadas durante la detección de imágenes relacionadas con las Bounding Box (BB). Para poder tratar una imagen es necesario saber descomponerla, y para esto, se enfoca como una matriz tridimensional que contiene datos analizables y fáciles de manipular.

Se procede a ver una serie de operaciones usadas para el tratamiento de la imagen, en relación a una serie de detecciones sobre el mismo objeto. La formación de estas es interpretable, y debido a esto, puede generar error, por lo que se necesitan tratar con una serie de funciones que perfeccionen las detecciones con las RRNN.

Intersection Over Union (IOU)

Gracias a este método se llega a ver si dos o más objetos colisionan. Se trata de obtener la cantidad de intersección que hay entre varias BB. Se utiliza para ayudar a Non-Maximum Suppression (NMS) a identificar dos o más BB que definen un mismo objeto, y con ello, evitar redundancia o falsos positivos.

La manera en la que se identifica es por medio de un porcentaje que define el encuentro de dos BB, tomando el total de la unión de ambas entre su intersección (ver Figura 3.14).

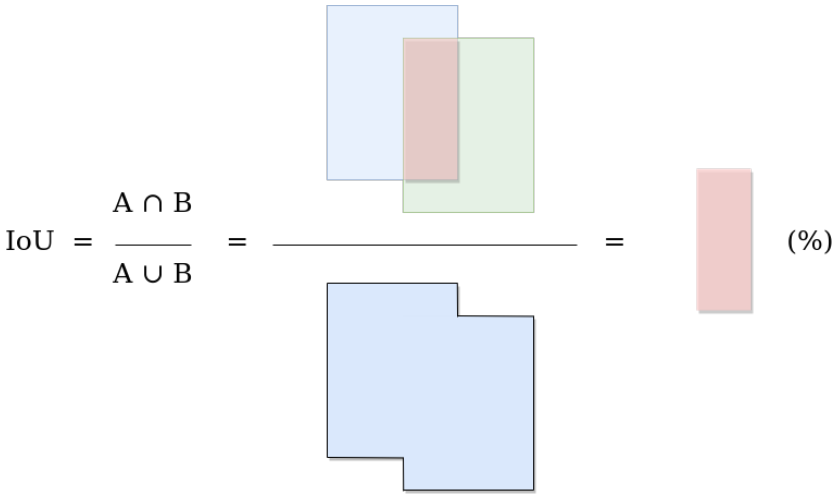


Figura 3.14: Función *IoU* representada gráficamente.

Non-Maximum Suppression (NMS)

Para comprobar la superposición de dos o más cajas, se ven cuales están próximas entre sí gracias a un umbral que define cómo de cercanas llegan a estar. Para ello, se utiliza la función ***IOU*** (ver Figura 3.14), la cual devuelve si dos o más objetos se intersectan por encima de un umbral.

Una vez se sabe si están o no superpuestos por encima de ese umbral, se guardan todas. Como se muestra en las tres figuras a continuación, se obtiene la siguiente conclusión según el threshold indicado: la supresión de uno de los elementos (falso negativo, Figura 3.15), la definición de todos los existentes (ver Figura 3.16) o por último, sin ningún filtro de detección, redundancia o falso positivo (ver Figura 3.17).

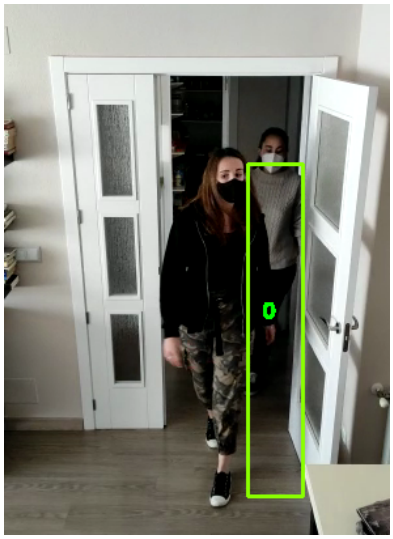


Figura 3.15: *NMS* con umbral de 0'3.

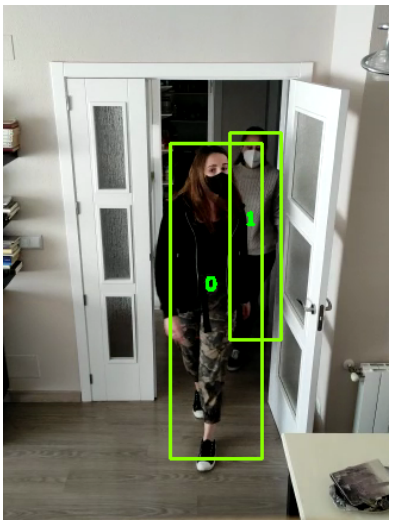


Figura 3.16: *NMS* con umbral de 0'6.

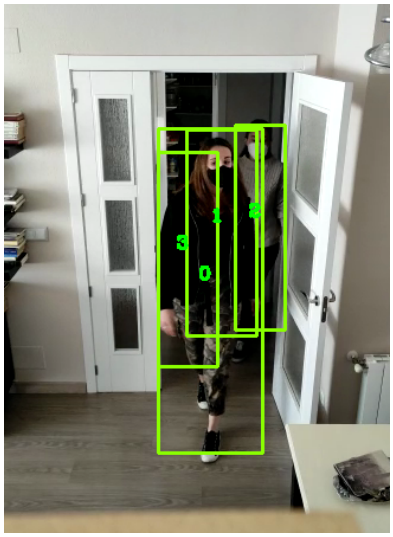


Figura 3.17: *NMS* con umbral de 1.

La manera en la que mejor se representarían objetos en colisión, estaría ajustado entre un valor de 0 a 1, donde una colisión nula sería el 1 y por lo tanto no se aplica el NMS. Las BB que prevalecerían serían las de mayor porcentaje de acierto, respecto a su clasificación.

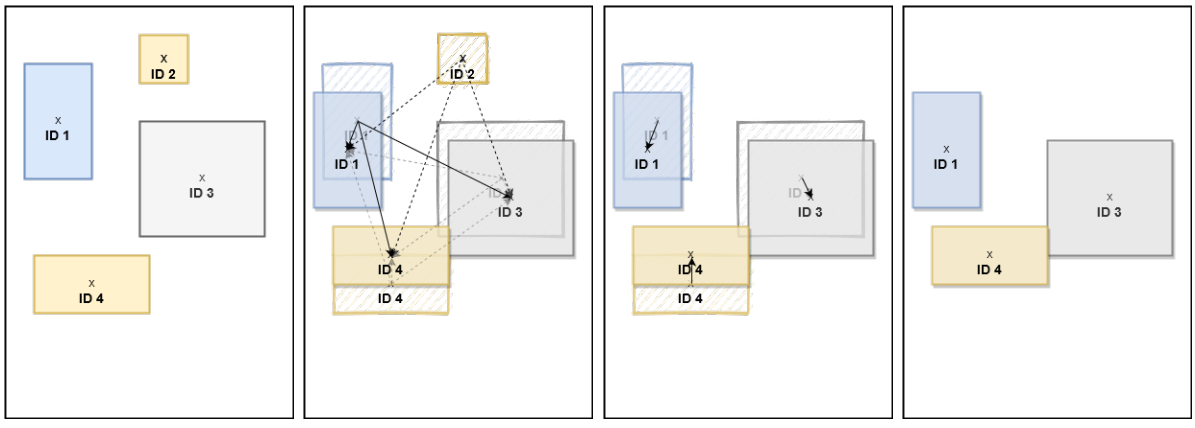
Centroid Tracker (CT)

Este concepto es muy útil a la hora de identificar el movimiento de una BB (caja delimitadora de objetos). Confiere una serie de utilidades, como la de hallar el centro de una figura a la cual se hace un seguimiento, o llevar un historial de ubicaciones y calcular su movimiento dentro de la matriz mediante una serie de algoritmos.

Usando un CT de una BB, se puede calcular la nueva posición del elemento en cuestión, y para ello se necesitan seguir cuatro pasos entre dos imágenes [19].

El concepto se fundamenta en llevar un historial de centroid trackers, que al pasarle una lista de BB, calcula los centros nuevos. Una vez determinadas las nuevas posiciones y las antiguas, se calcula la distancia Euclídea desde los centros antiguos a los nuevos. A partir de este punto, se puede saber cuál es la posición más cercana de los centros anteriores.

Como se representa en la Figura 3.18 apartado *(c)*, si un centro antiguo no tiene un nuevo centro asociado, se elimina el rastro de este. Por lo contrario, si es uno nuevo el que no tiene uno anterior asociado, se creará un nuevo registro.



(a) (b) (c) (d)

Figura 3.18: (a) Paso 1. (b) Paso 2. (c) Paso 3. (d) Paso 4.

Teniendo toda esta información en cuenta, se ha estudiado cómo se pueden detectar objetos y seguir su rastro. Por tanto, se puede crear un sistema de detección que mire si un objeto entra o sale de un límite (contador de ocupación), o bien, una serie de BB siguen un flujo interpretable (mapa de calor).

En este apartado se expondrán los dos tipos de sensores que se han tenido en cuenta para realizar este proyecto. Se procederá a explicar cómo funcionan, cómo miden y cuáles son sus límites.

Capítulo 4

Explicación del modelo

4.1. Diseño estructural completo

Esta sección está dedicada a la estructura completa del modelo realizado. Se explicarán sus elementos y comportamiento, para más tarde formar una estructura con la que poder hacer pruebas.

4.1.1. Aplicación Web

Para facilitar el acceso a las diferentes pantallas de la aplicación, en la parte superior se puede encontrar una barra de navegación (ver Figura 4.1). Se subdividirá el contenido de este apartado, de forma que se explicarán las funcionalidades de la web, como si por este menú se navegase.

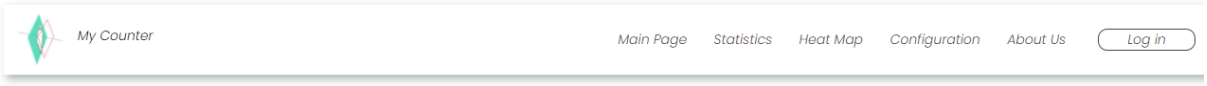


Figura 4.1: Barra de navegación.

- Página Principal

  Se accede a ella por defecto al abrir la aplicación web (ver Figura 4.2). Además, es pública, es decir, cualquiera que acceda a la web vía Uniform Resource Locator (URL) puede consultar la ocupación de aforo del local en el momento actual. Para ello, se incluyen un gráfico indicador, una tabla y una imagen. Todos ellos tienen asociado un comportamiento AJAX, acorde al tiempo en el que se reciben los mensajes del broker. Este comportamiento es el responsable de actualizar la vista.

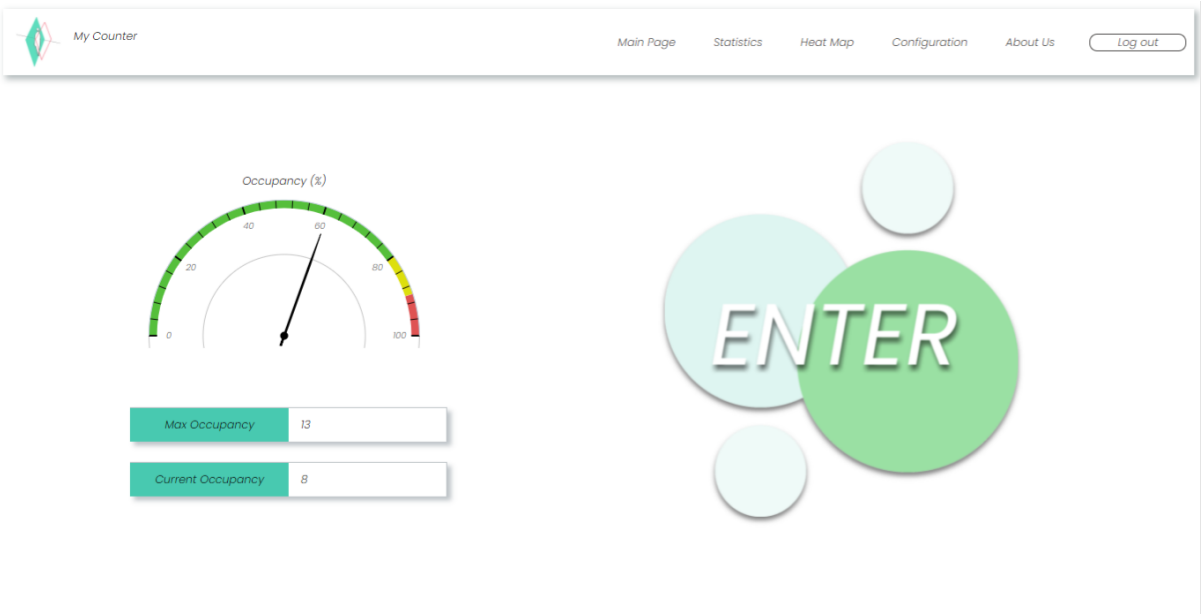


Figura 4.2: Página Principal (Aforo Medio-Alto).

Concretamente, el gráfico indica la ocupación actual, expresado en porcentaje. La tabla nos muestra dos filas; la superior, en la que se puede ver el aforo máximo (configurado por el administrador), y la inferior, que muestra cuantas personas están en el interior del recinto. Por último, aparece una imagen que cambia su apariencia en caso de llegar al máximo aforo, y así, indicar que es necesario esperar hasta que alguien salga (ver Figura 4.3).

Figura 4.3: Página Principal (Aforo Máximo).

- Estadísticas

  Además de mostrar datos en tiempo real, se llegó a la conclusión de que tendría más utilidad si hubiese algún tipo de persistencia y análisis de los datos obtenidos. Es por ello, por lo que se implementó una página con dos tipos de gráficos accesibles para usuarios de tipo administrador.

  En el primero de ellos se refleja el aforo del recinto durante el día actual en intervalos de diez minutos (ver Figura 4.4).

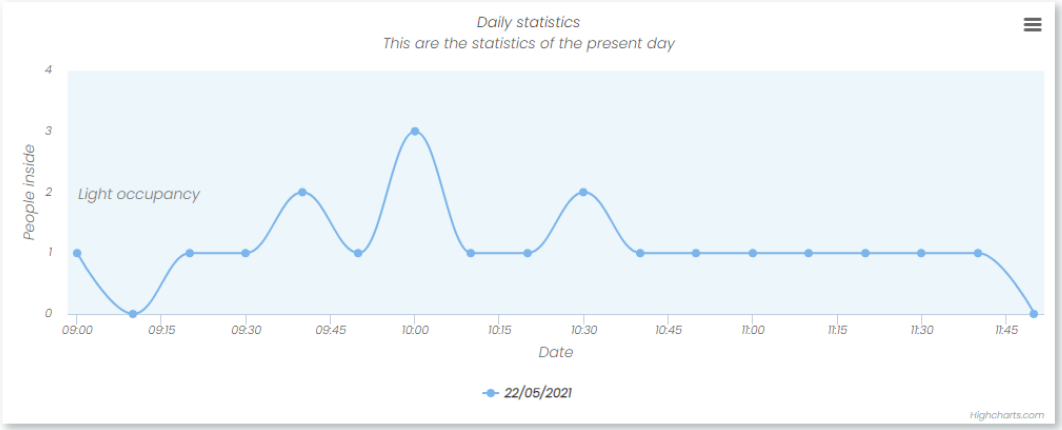


Figura 4.4: Gráfica Diaria.

En el segundo, se recuperan desde la base de datos las lecturas de ocupación de los últimos siete días, realizadas también en intervalos de diez minutos, y se muestran de forma que cada serie corresponde a un día (ver Figura 4.5). Es posible que, como en este ejemplo, aún no se hayan almacenado datos relativos a la última semana, en ese caso se mostrarán los datos de los que se disponga hasta la fecha.

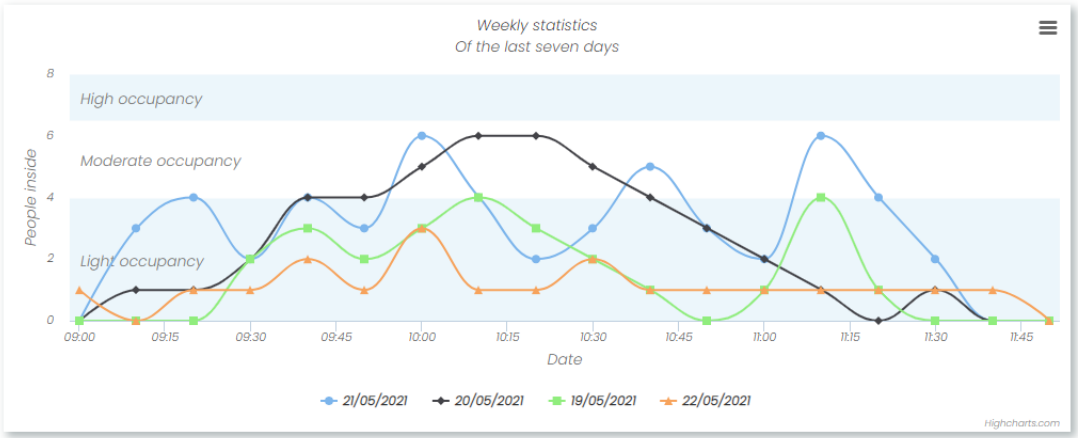


Figura 4.5: Gráfica Semanal.

Para esta tarea se ha empleado la librería Highcharts[1], que es una de las bibliotecas de gráficos basadas en estándares para dispositivos web, móviles y de IoT más usadas del momento. Ofrece, junto a los gráficos, varias opciones para exportarlos. De esta forma, si se quiere, pueden ser guardados y almacenados para futuras estadísticas o revisiones de resultados obtenidos.

- Heatmap

  Para la parte referente a la pestaña del mapa de calor, se consideró que la mejor forma de representarlo para su fácil comprensión, sería utilizando un gradiente de color. Para ello, se superpone sobre la imagen, una matriz de celdas y se aplica un gradiente sobre cada una, de forma que, a partir del máximo valor leído, se define el resto. El tamaño de cada celda coincide con el tamaño definido en el publicador.

  Concretamente, se tiene que un color azulado o más frío nos indica que por esa zona se circula con poca frecuencia, y uno cálido (más cercano al rojo) representa una zona más concurrida (ver Figura 4.6).

Figura 4.6: Gama del gradiente.

Por último, se entiende que no se quieren ver representadas las casillas que no contengan información relevante, por ello, se les modifica la opacidad para no mostrarlas en el gradiente. Para dar cohesión, además, se impone un desenfoque. El resultado final se ve reflejado en la Figura 4.7.

Figura 4.7: Representación Mapa de calor.

[1] https://www.highcharts.com/

- Configuración

  Se entiende como configuración de esta aplicación aquellos valores que son susceptibles de cambiar dependiendo del entorno en el que se despliegue. Estos son: aforo máximo; URL e ID del broker y URL, usuario y contraseña de la base de datos.

  Estos datos están planteados de modo que se guarden en un archivo JavaScript Object Notation (JSON), y se carguen desde la página de configuración de la web (ver Figura 4.8) para poder ser editados en cualquier momento por un usuario de tipo administrador.

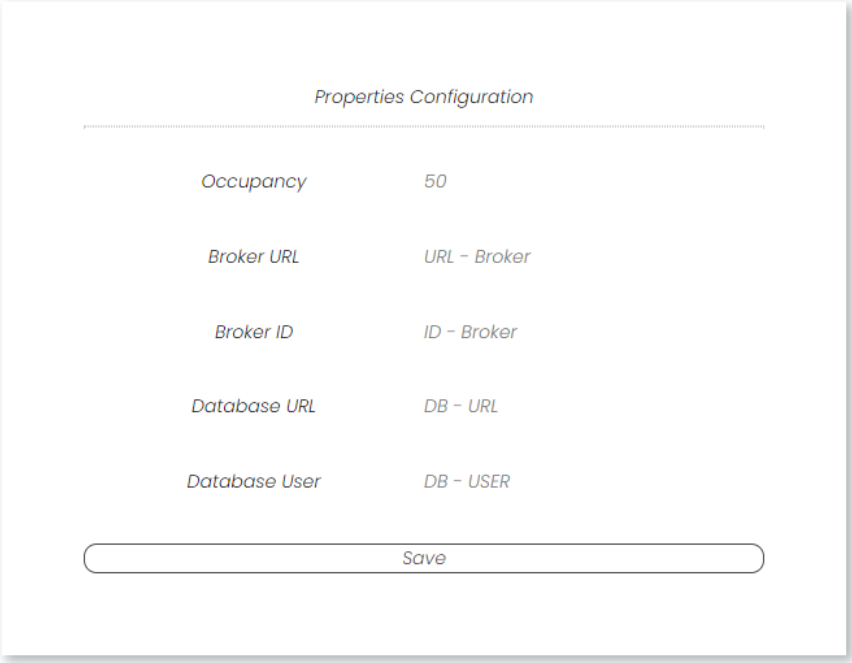


Figura 4.8: Panel Configuración.

El fichero de configuración se carga al intentar acceder a la web. En caso de no conseguir leerlo, el usuario será redirigido a una pantalla donde se le avisará del error y se le ofrecerá la opción de reintentar leer el fichero y volver a la página principal (ver Figura 4.9).

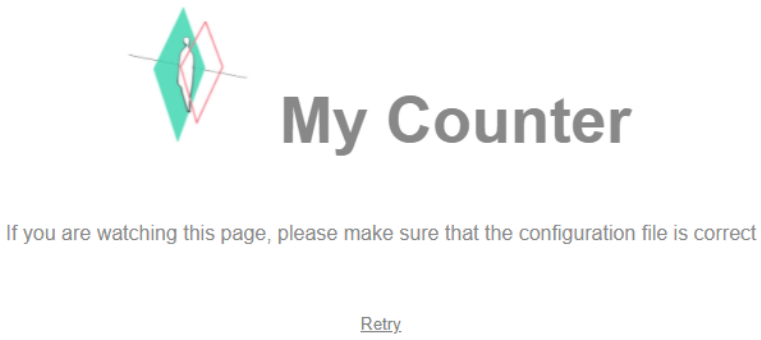


Figura 4.9: Página Error.

Desde la pantalla de configuración, es posible también parar o volver a establecer la comunicación con el broker en caso de ser necesario (ver Figura 4.10).

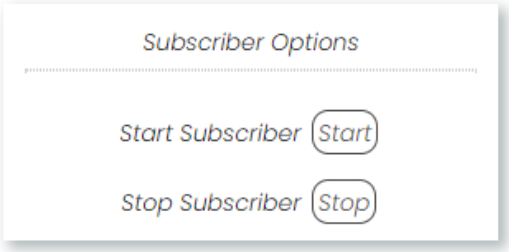


Figura 4.10: Opciones Broker.

Además añadir, que puesto que la web está enfocada a que haya un administrador con más permisos que el resto, se ha implementado un panel (ver Figura 4.11) que permite editar tanto el nombre de usuario como la contraseña del mismo.

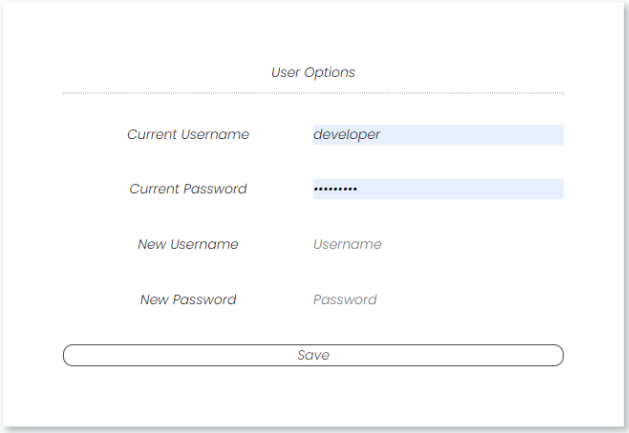


Figura 4.11: Edición Usuario.

- Login

  Utilizando las estrategias de autorización mediante anotaciones de Wicket [11], todas las páginas de la web, exceptuando la principal, tienen acceso protegido. Esto implica la creación de un formulario de login que aparece cada vez que se intenta acceder a ellas (ver Figura 4.12). Una vez introducidos los datos, se comprobará si corresponden a algún usuario dado de alta en la base de datos y se continuará con el acceso a la página seleccionada. Además, se guardará la sesión y el usuario podrá navegar libremente.

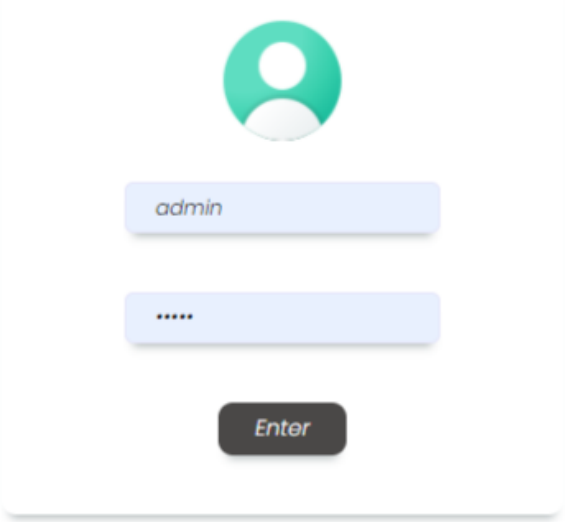


Figura 4.12: Página Login.

Una vez aclarado el funcionamiento de la web, se puede proceder a la justificación del diseño. Visualmente, se ha mantenido una estética en tonos blancos, verdes y azulados. Estos colores ayudan a evocar profesionalidad, seguridad, higiene visual y claridad [20, 21], entre otras, en el usuario. Por este mismo motivo, para ayudar a no saturar la vista o tener demasiados estímulos visuales que impidan la correcta visualización de los datos, se ha mantenido una distribución simple de los elementos dentro de cada pantalla.

Base de datos

La base de datos está compuesta por tres tablas de tipo InnoDB (ver Figura 4.13).

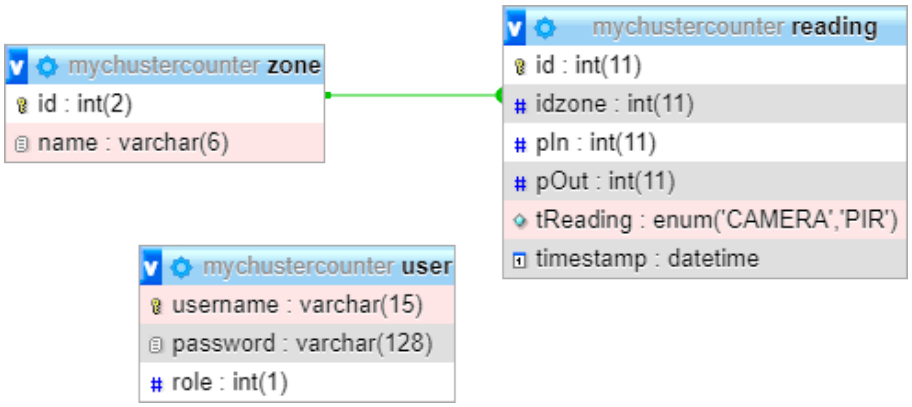


Figura 4.13: Esquema de la base de datos.

Tabla user

Para el acceso a las funciones de configuración, se necesita tener un usuario registrado que deberá estar almacenado en esta tabla. Sus campos son los siguientes:

- *username*: es un identificador único (clave primaria), que se corresponde con el nombre de usuario con el que se inicia sesión en la web.
- *password*: cada usuario para poder loguearse tiene una contraseña, la cual se guarda de manera cifrada empleando el algoritmo *SHA-512*.
- *role*: campo que almacena el rol que tiene el usuario. Si es un *0*, quiere decir que tiene permisos de administrador; de lo contrario, tendrá un *1*.

Tabla zone

Dado que el recinto puede disponer de una o más zonas de detección, estas necesitan estar almacenadas. Esta tabla está compuesta por los siguientes campos:

- *id*: id asignado a la zona. Es una clave primaria autogenerada.
- *name*: nombre descriptivo que identifica la zona (lugar de acceso al recinto).

Tabla reading

En esta tabla se almacenan los registros de ocupación (independientemente de la zona). Está compuesta por los siguientes campos:

- *id*: clave primaria, autoincrementada con cada nuevo registro que se persiste.
- *idZone*: clave externa que hace referencia a la zona del local (de la tabla "id-zone") en la cual se haya contabilizado el conteo de personas.
- *pIn*: número de personas que entran al local.
- *pOut*: número de personas que salen del local.

- *tReading*: enumerado que hace referencia al método por el cual se ha obtenido la información que se almacena. Tiene dos valores únicos (actualmente): "CAMERA", si se ha obtenido a través de una cámara (independientemente del tipo que proceda); o "PIR", si se ha obtenido a través de un sensor infrarrojo pasivo.
- *timestamp*: marca de tiempo en la que se han contabilizado los datos.

4.1.2. Conexión entre módulos

Debido a la necesidad de enlazar la aplicación de detección mediante inteligencia artificial, la detección mediante sensores y el mapa de calor con la aplicación web y gracias a las ventajas que nos otorga un broker como Mosquitto, se han desarrollado diferentes módulos adecuándolos a las acciones propias de cada una de las aplicaciones. Estos módulos cuentan con las siguientes características:

Publicadores

En el caso del módulo del publicador principal, que se encarga de enviar los datos obtenidos a partir de la detección de vídeo, es importante desarrollar un sistema sencillo que sea capaz de conectarse al broker y reconectarse en caso de caída. Una vez establecida la conexión, se encargará de crear el mensaje con una estructura que definiremos más adelante, y finalmente publicar el mensaje para que los suscriptores puedan recibirlo y tratarlo.

Se ha empleado la librería oficial de Mosquitto, llamada Libmosquitto, que permite desarrollar aplicaciones tanto en C como C++.

La aplicación del mapa de calor además de contar con las características anteriormente mencionadas, implementa el envío del primer fotograma que procesa. Esto se debe a que desde el punto de vista de la aplicación web, es necesario mostrar el mapa de calor con una imagen de fondo que represente la zona observada. Esta imagen no es necesario codificarla, ya que al solo enviarse una vez no supone ningún problema en la red.

Suscriptor

Una vez se ha explicado el módulo del publicador, se va a explicar el del suscriptor. Este debe contar con aún más robustez que el publicador, ya que un fallo en el broker no puede dejar inaccesible la web. Por ello, se implementan métodos para conectarse al broker automáticamente en caso de caída.

Por otro lado, este módulo debe ser capaz de recibir, procesar y guardar todos y cada uno

de los mensajes de los publicadores a los que se encuentra suscrito. Por esto, se utiliza una cola de mensajes en la cual se van almacenando según van llegando, para posteriormente ser procesados y almacenados los datos. Los diferentes mensajes son:

- Mensajes con el número de personas que entran y salen.
- Primera imagen del vídeo de detección de calor.
- Matriz con los datos del mapa de calor.

Para implementar todo este módulo se utiliza la librería Paho, implementada por la fundación Eclipse y que se utiliza para desarrollar aplicaciones en Python (PY), Java o C++.

Características comunes

Codificación de los mensajes

A la hora de realizar los diferentes módulos se han pensado y probado diversas codificaciones de mensajes. Primero de todo, se han enviado tres mensajes desde la aplicación de detección. Estos mensajes eran el total de personas entrantes, el total de personas salientes y la fecha. Pero tras realizar pruebas, y sobretodo, al intentar mejorar la escalabilidad del sistema, se ha observado que la saturación de la red y la dificultad de implementación de los algoritmos del módulo del suscriptor podrían aumentar.

Por todo esto, se ha creado una codificación con la que se puede enviar toda la información necesaria en un único mensaje y además es más sencilla, ya que el protocolo MQTT está pensado para envío de mensajes simples (ver Figura 4.14 y Figura 4.15).

Figura 4.14: Ejemplo mensaje Aplicación de Detección.

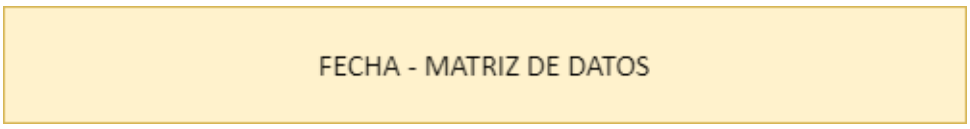


Figura 4.15: Ejemplo mensaje Mapa de Calor.

Una vez este mensaje llega a la Aplicación Web, esta se encarga de procesarlo y almacenar el dispositivo y los datos que ha recibido, para posteriormente utilizarlos en las diferentes gráficas.

Estructura de los Temas

Al realizar el cambio en la codificación de los mensajes, se crearon unos temas que permiten escalar la aplicación y que además, utilizan la estructura de árbol que MQTT proporciona (ver Figura 4.16).

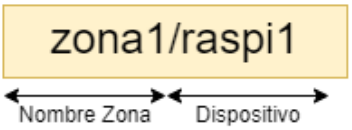


Figura 4.16: Ejemplo de tema utilizado.

Gracias a esta estructura de temas de zonas, se consigue que pueda haber diferentes dispositivos en una misma zona, e incluso diferentes zonas en un mismo recinto (ver Figura 4.17).

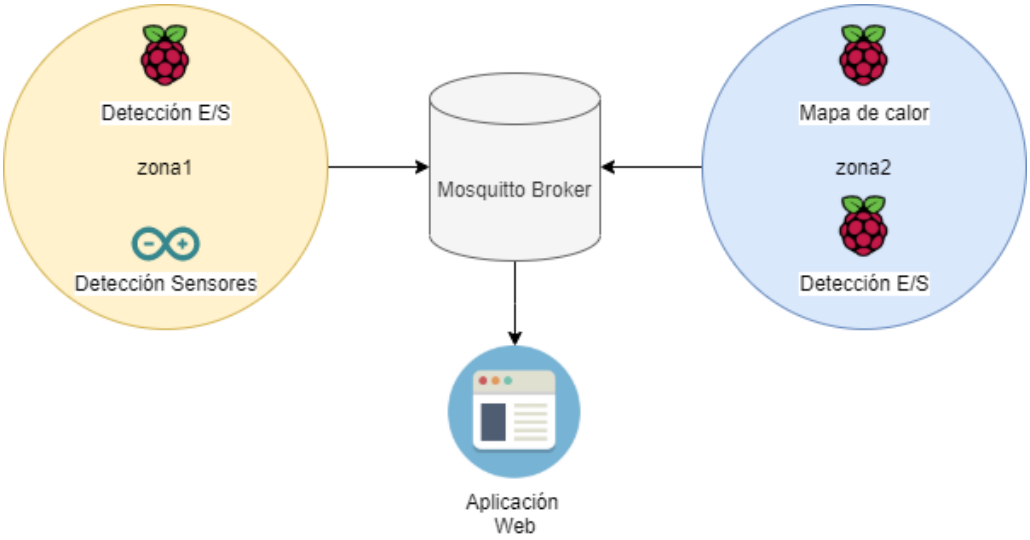


Figura 4.17: Ejemplo con múltiples zonas.

Inicio de sesión en el broker

Debido a la configuración que se ha realizado en el broker, la cual se explicará más adelante, ha sido necesario implementar un inicio de sesión automático en los diferentes módulos de las aplicaciones. Esto se realiza de una manera muy sencilla gracias a las librerías utilizadas en cada módulo.

Conexión y reconexión

Por último, se observó la necesidad de que la aplicación web pudiera seguir funcio-

nando en caso de que el broker sufriera algún fallo que lo dejase inhabilitado. Por ello, se implementó un sistema de reconexión automática y para el peor de los casos un sistema manual. Gracias a esto se logró que la web pudiera seguir activa para la consulta de las diferentes estadísticas. En las aplicaciones de detección de vídeo y detección por sensores, también se ha implementado este sistema de manera que se reconecte al broker cuando esté activo y pueda seguir enviando datos.

Configuración del Broker

Debido a la necesidad de hacer que el sistema de intercambio de mensajes sea privado, es necesario añadir una configuración adicional tras la instalación del broker, mediante la cual, se selecciona un usuario y una contraseña que deben usar todos los dispositivos que requieran conectarse al broker. De esta forma, se consigue una mayor privacidad y evitar conexiones de dispositivos desconocidos. En esta ocasión, se ha decidido no utilizar el sistema TLS que también se puede configurar en el broker, ya que los datos que se envían no necesitan ese nivel de protección.

4.1.3. Detección de Ocupación

La detección de ocupación trata de comprender cómo capturar la información y transformarla en una serie de datos interpretables por un ser humano. De esta manera, se entiende que, esta parte del modelo es la parte del publicador de datos.

La fracción de detección de ocupación sobre imágenes está desarrollada gracias a las herramientas de detección y redes neuronales descritas anteriormente. Para ello, se utilizará una red neuronal ya entrenada para TFL, la red neuronal es de tipo SSD Lite y se le ha aplicado Mobilenet V3 entrenada con *COCO14*. Sus principales características son:

- Identifica *90* tipos de clases de objetos.
- El tamaño de imágenes de entrada es *320×320*.
- La función de activación que ha sido utilizada fue Función Lineal Rectificadora (ReLU) *6*.
- El umbral del IOU para la salida es de *0.6*.
- El porcentaje de acierto por cada objeto se le tiene que aplicar Función Sigmoide (SIGMOID) para poder interpretarlo.
- Para evaluar su eficacia se han comprobado con *8000* ejemplos.
- Adaptada a la API de TFL con modelos .tflite.

La otra fracción es la detección de ocupación con sensores, en el que se van a utilizar diferentes tipos, para disponer cuál es el más preciso en cuanto a la detección de entrada y salida. Se hará un breve análisis de cómo se sitúan (físicamente) y qué alcance tienen.

Percepción de Entrada y Salida

Esta parte del modelo se entenderá como una detección en la zona de E/S del recinto cerrado, donde pasado un umbral, cuenta si cada individuo detectado se encuentra en la parte interior del recinto. La manera en la que puede recabar imágenes es mediante un vídeo ya grabado (para hacer pruebas), a través de una webcam conectada directamente a la RPI o una cámara IP conectada a internet.

La idea se basa en recorrer cada frame y hacer las detecciones pertinentes, para luego mejorar los resultados en la parte de NMS y por último las BB seleccionadas serán las utilizadas para calcular el CT. Sabiendo qué recorrido hicieron los elementos guiados, se puede saber si estos entran o salen y llevar la cuenta de los mismos.

Observando la Figura 4.18, se puede ver cómo ambas personas cruzan una línea de arriba a abajo. Así, si sabemos definir cuál es la salida y cuál es la entrada, se logra saber por cada centroide[2] si entra o sale del recinto. La línea roja es la que delimita el umbral que se debe cruzar para contar con la persona dentro del recinto cerrado, para poder incluir la cuenta, debe estar dentro del perímetro de las líneas verdes.

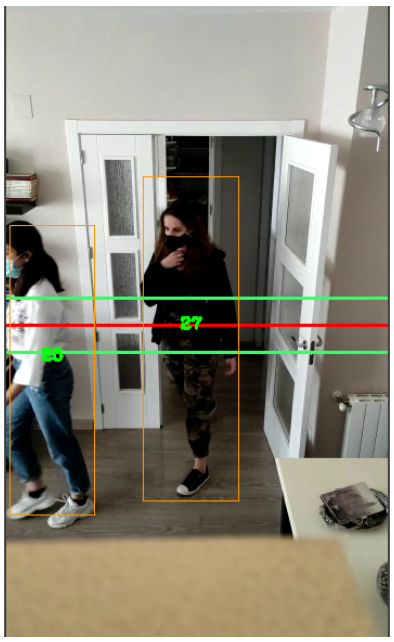


Figura 4.18: Detección E/S.

[2]Centro de simetría en una figura geométrica.

Para obtener la mejor definición del modelo, se tuvo que prestar mucha atención en cada uno de los módulos, siendo el de la detección de las BB el más importante. Todos los datos generados serán enviados para su posterior estudio, y de esa forma, estarán cubiertas todas esas áreas de la zona o zonas donde se delimite el estudio. Será necesario por tanto, una fuente de visualización que pueda tramitar el programa. Más adelante se tratará parte de la mejora de este modelo y su posterior análisis.

La manera de situar el dispositivo para la captura de imágenes comienza definiendo correctamente qué se considera dentro y fuera para un análisis adecuado. Comúnmente, el interior del recinto se fija a partir del límite coincidente con el que se define la zona, el problema es que no siempre coincide con una entrada. El modelo, por su parte, definirá si este se ha cruzado. Un ejemplo muy visual es el de la Figura 4.19, que muestra cómo se puede tener en cuenta la posición del dispositivo mientras definamos un interior (amarillo) y un exterior (rojo).

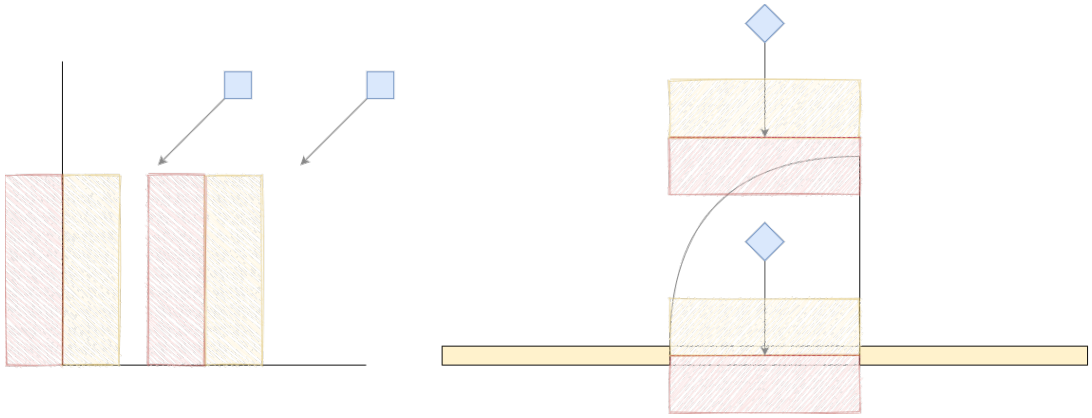


Figura 4.19: Puntos de percepción E/S.

Mapa de Calor

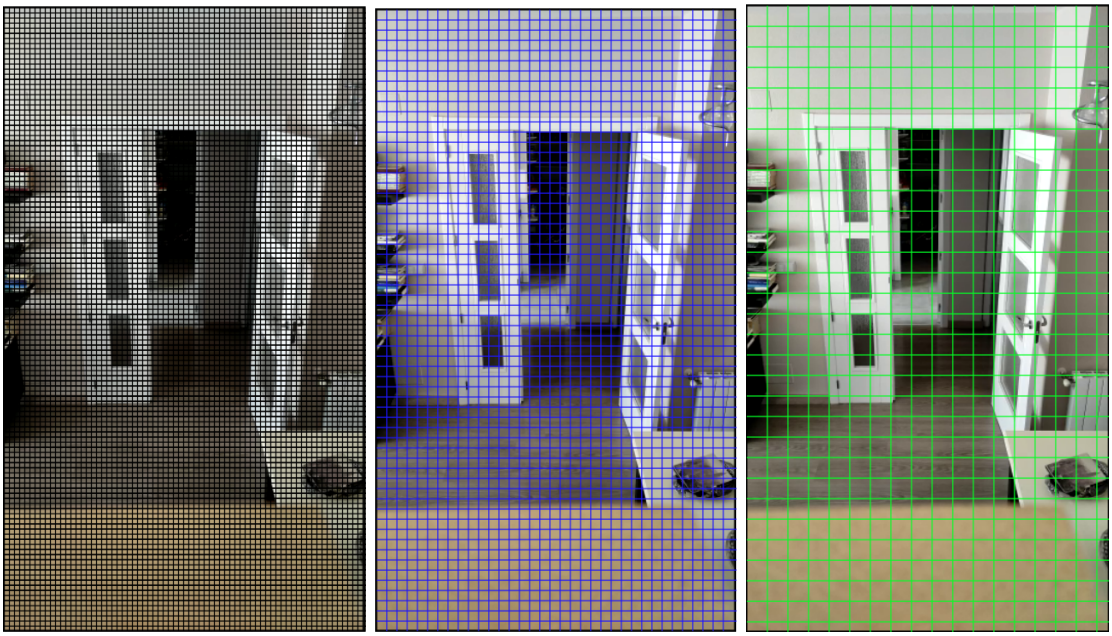
Para poder percibir mejor el comportamiento dentro de estas zonas descritas, es necesario saber cómo se comportan los individuos de manera interna. Por esto mismo, se ha implementado en el modelo un mapa de calor, que permite estudiar diferentes parámetros que luego se puedan analizar de manera visual.

La importancia y la precisión de los datos recogidos en cada fotograma, determinan no sólo la cantidad de información, sino también cuántos datos serán enviados a través del broker a la web, ocupando el ancho de banda. Esta es la razón por la que el programa que recoge el mapa de calor está pensado para modificar el tamaño de sus celdas y así tener cubierta esta especificación.

Cada una de las imágenes que se representa en el modelo de la Figura 4.20, tiene exactamente 352×640 píxeles, en los que se podría situar el centroide de un objeto. A la

hora de enviar los datos a través de un broker, o bien, que lo procese una web, se ha de tener en cuenta la necesidad de transmitir una matriz de *225.280* celdas, y por tanto, el tamaño del mensaje será proporcional al número de celdas enviadas.

Se ha pensado aplicarle un tamaño de celda fijo, con lo que se englobaría más píxeles en un menor número de celdas. El problema de esta solución para evitar la saturación es, que al reducir la matriz, también se reduce la exactitud de los datos obtenidos. Por ello, habrá que probar de qué manera se obtienen más datos evitando la congestión. La manera en la que se insertan los datos se basa en tomar la posición del centro e introducirla en una matriz de este modo.



(a) (b) (c)

Figura 4.20: Tamaño de celda. (a) 5 px (b) 10 px (c) 20 px.

Gracias a esta sencilla operación, se puede modificar el tamaño de cada celda sin necesidad de ningún cálculo y casi sin coste adicional.

```
for cada uno de los centros seleccionados do
   Insertar mapa[pos. x / tamaño celda][pos. y / tamaño celda] = valor
end for
```

Los datos se recogen en una matriz bidimensional cuyas posiciones se actualizan gracias al CT tras cada fotograma leído. Así se envía de forma sencilla al suscriptor. Posteriormente, al aplicarle un gradiente se puede interpretar de forma visual. Viendo la Figura 4.21, se refleja que las zonas más calientes son representadas en torno a un valor máximo. Enviando este, junto a la matriz y la imagen de fondo, se podría representar cómo interactúan ciertos individuos en una zona.

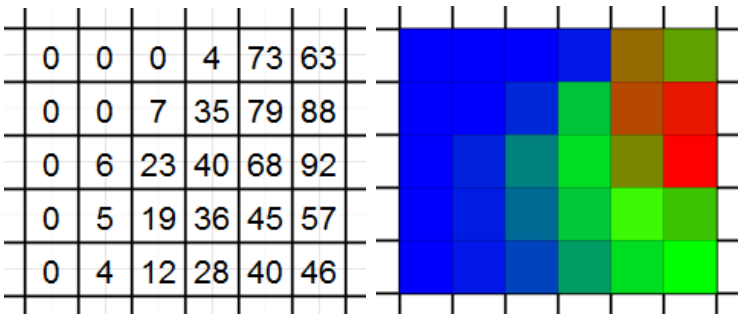


Figura 4.21: (a) Número por celda. (b) Traducción Gradiente.

La manera en la que se sitúa el dispositivo de captación de imagen consiste en ver cómo incluir más espacio, o bien, qué puntos interesa capturar. Como se puede ver en la Figura 4.22, una vez definida una zona cerrada habrá que tener en cuenta qué rango de visión abarca el dispositivo. Una vez establecido el primer punto, habrá que ver si es pertinente capturar una cantidad grande de espacio, o bien centrarse en un punto concreto, como en el ejemplo el dispositivo rojo.

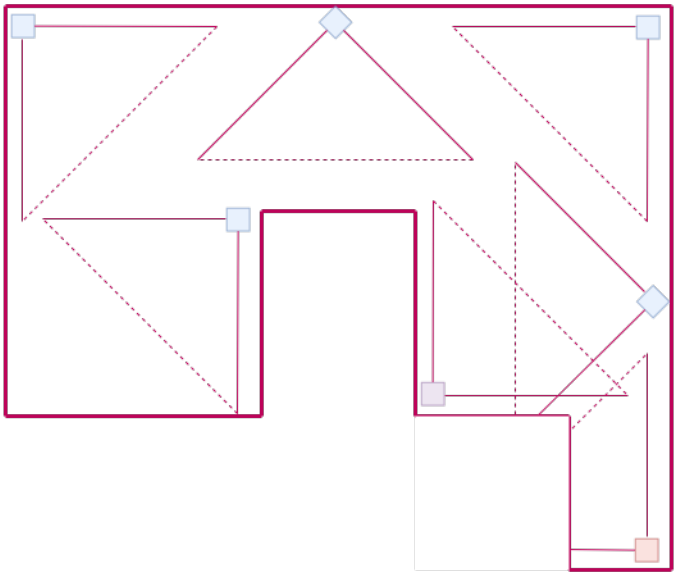


Figura 4.22: Puntos de percepción HeatMap.

Sensores

A continuación se procede a definir el modelo realizado para el correcto funcionamiento de los sensores. Para ello, y ante futuras pruebas que se realizarán, se comentará cómo y

dónde se han ubicado los sensores, además de las limitaciones que puedan tener.

PIR

Como se ha visto en la Sección 2.1, el sensor piroeléctrico basa su funcionamiento en el hecho de captar la radiación infrarroja emitida por las personas, animales u objetos que pasen por delante del rango de detección que tiene. Para el modelo se ha utilizado, en concreto, el sensor HC-SR501. Se ha de destacar que puede detectar hasta un rango de *6/7* metros, con un ángulo efectivo de 110^{o}. En la Figura 4.23 se pueden ver definidas las siguientes partes de este sensor:

- (1) Ajusta el tiempo en el que mantiene el dato.
- (2) Ajusta la longitud máxima que detectar.
- (3) Conexión 5V, Salida de datos y GND.

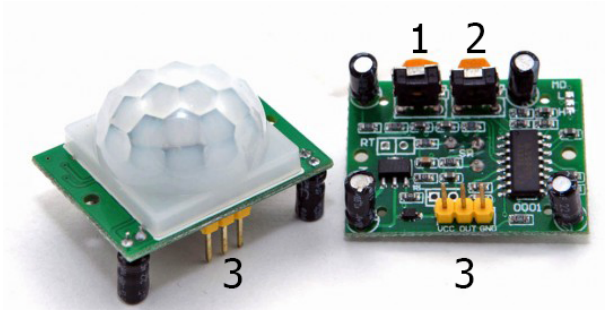


Figura 4.23: Partes del PIR.

Si bien es cierto que este sensor puede detectar a una persona, puede ser que no cumpla el cometido de este proyecto, que es el conteo de personas en un espacio cerrado. Esto se debe a que en el caso de que el recinto tuviera entradas y salidas diferenciadas (es decir, accesos unidireccionales y que se garantice que así sea y se cumpliera), no habría problema, ya que con un único sensor en cada lugar sería suficiente para determinar si entra o sale. En la Figura 4.24 se puede ver cómo el sensor realiza la detección de una persona.

En cambio, la mayoría de accesos de los recintos son bidireccionales. Es decir, las personas pueden entrar y salir por la misma puerta. La solución propuesta es la de colocar dos sensores juntos: el sensor *1* es el más cercano a la calle y el sensor *2* el más alejado. Pues bien, si una persona va a entrar, en primer lugar el sensor *1* recibirá los datos correspondientes, acto seguido el *2* recibirá los mismos datos. Cuando la persona haya avanzado un poco más, el sensor *1* dejará de recibir datos y posteriormente, lo hará el *2*. De esta manera se sabe que una persona está entrando (y este proceso a la inversa significaría la salida de alguien). Es decir, con la

desactivación del segundo PIR se conoce la dirección del individuo detectado (ver Figura 4.25 y Figura 4.26).

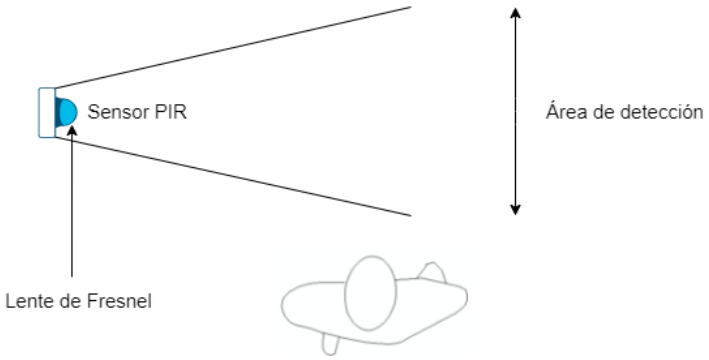


Figura 4.24: Esquema de detección del sensor PIR.

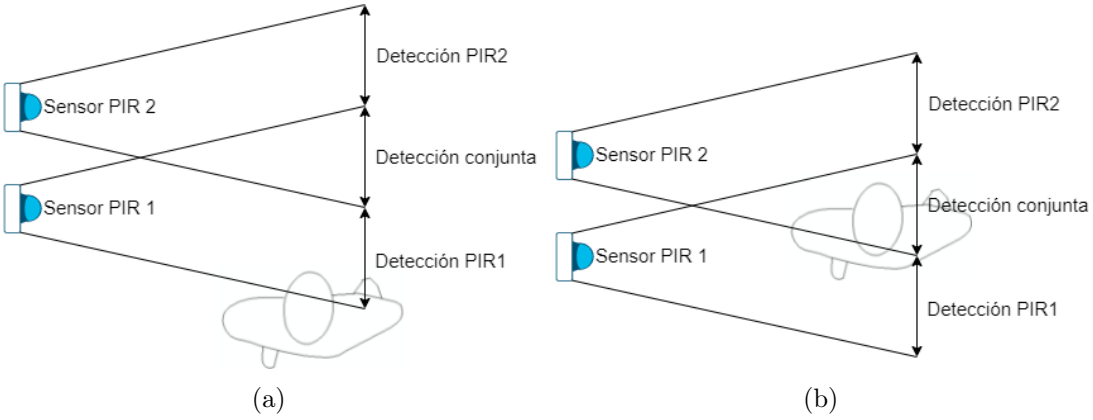


Figura 4.25: (a) PIR1 detecta (b) PIR2 detecta.

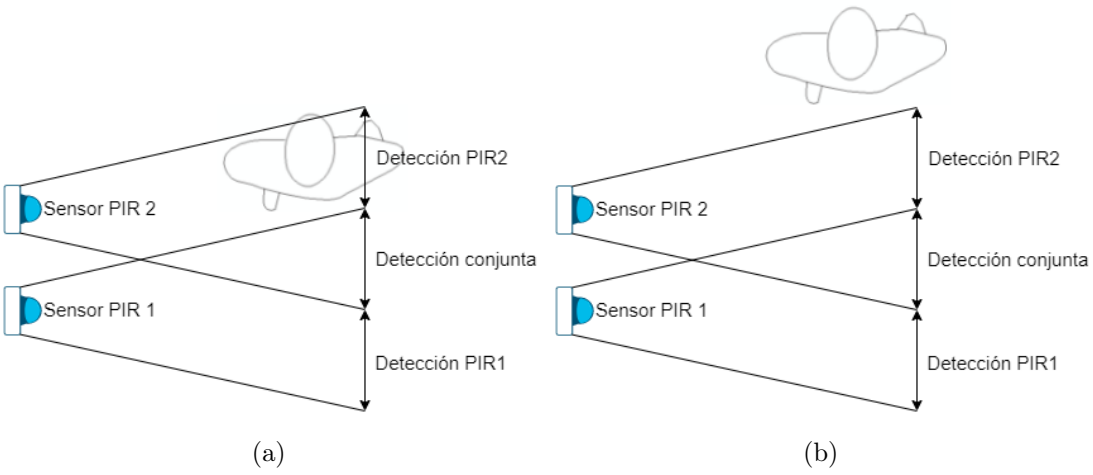


Figura 4.26: (a) PIR1 deja de detectar (b) PIR2 deja de detectar.

Por esto, dado que la mayoría de recintos tienen accesos bidireccionales, para poder llegar a un mayor número de recintos se ha elegido desarrollar el modelo con dos sensores PIR. La disposición de dichos sensores ha de ser la que se puede ver en la Figura 4.27, uno dentro al pasar la puerta, y el otro antes de pasar por ella.

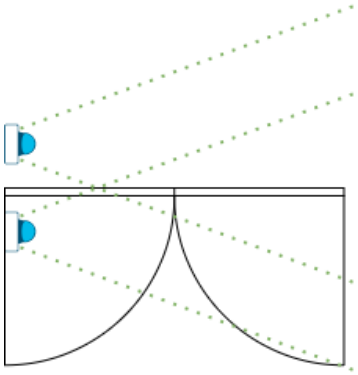


Figura 4.27: Lugar en el que se encuentran los sensores.

También se necesitará un módulo NodeMCU como se puede ver en la Sección 2.1, siendo utilizado el NodeMCU V3 dado que tiene una toma de *5*V, que es la que necesita el sensor PIR. La placa recogerá la información captada por ambos sensores y será enviada a través del protocolo MQTT a un broker.

Ultrasonidos

Como ya se explicó en la Sección 2.1, el sensor ultrasónico basa su funcionamiento en el envío de una señal ultrasónica por uno de sus transductores, mientras que el segundo se mantiene a la espera de recibir el rebote de las ondas por la colisión con un obstáculo. Para este modelo, se ha utilizado el sensor HC-SR04 (ver Figura 4.28), el cual puede abarcar hasta *4* metros de distancia de manera teórica, aunque en la práctica puede verse reducido a la mitad (*2* metros). El ángulo efectivo es de unos 15^{o} y la frecuencia de pulso es de *40*KHz. No se puede determinar si lo detectado es una persona o un objeto.

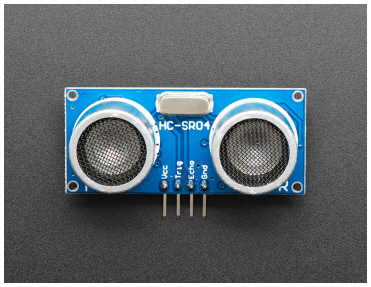


Figura 4.28: Ejemplo de un sensor ultrasónico.

Como se puede apreciar en la Figura 4.28, este sensor tiene cuatro conexiones:

- **VCC y GND:** corriente de 5V y tierra.
- **Trigger:** pin disparador del ultrasonido.
- **Echo:** pin receptor del ultrasonido rebotado.

Se parte de que la velocidad, en este caso, 340 m/s por ser la del sonido, es la distancia recorrida (dos veces la distancia al objeto) partido por el tiempo que se tarda en recorrer dicha distancia. De manera visual, se muestra a continuación la fórmula aplicada al controlador del sensor ultrasónico.

$$\frac{340m}{s} \times \frac{1s}{1000000us} \times \frac{100cm}{1m} = \frac{2d}{t}$$

$$d(cm) = \frac{t(us)}{59}$$

Gracias al funcionamiento del sensor, se puede conocer la distancia a la cual ha detectado un objeto o una persona. Siempre y cuando en un acceso se establezca que una parte del mismo es de entrada y la otra de salida del recinto (como se puede ver en la Figura 4.29), y dicho flujo sea respetado, en función de la distancia proporcionada podremos conocer si una persona está entrando o saliendo del recinto.

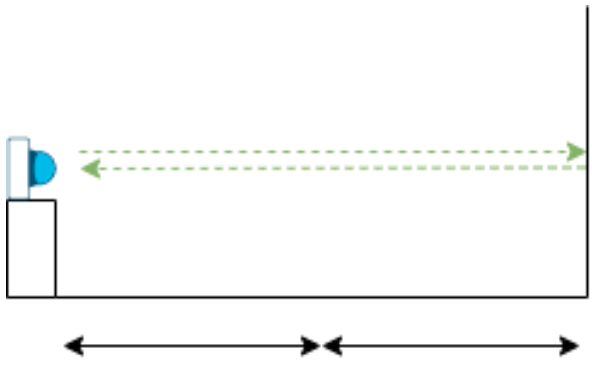


Figura 4.29: Esquema de un acceso bidireccional.

Al igual que el caso de los sensores PIR, será conectado a un módulo NodeMCU, en este caso el V3, que envía los datos a un broker.

Conexión de los sensores a la placa NodeMCU

En este apartado se mostrará de manera esquemática y se explicará cómo fueron conectados los sensores a la placa utilizada, en este caso, NodeMCU V3 (ver Figura 2.8).

PIR

A la hora de conectar los sensores PIR, se debe realizar como se indica en la Figura 4.30.

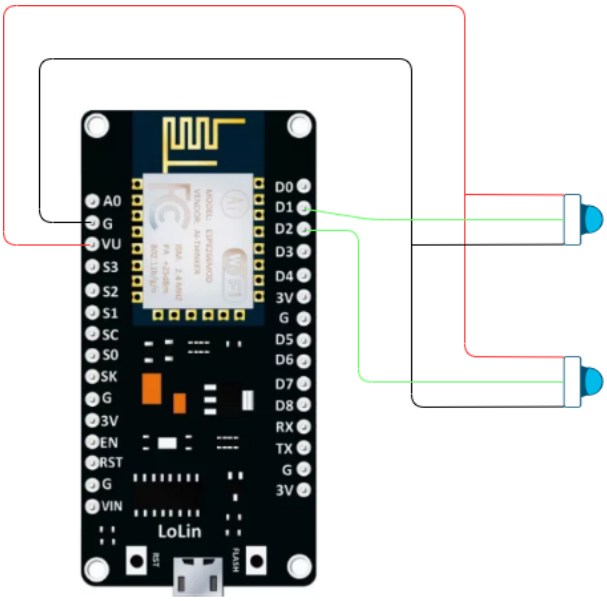


Figura 4.30: Esquema de la conexión de los sensores PIR.

- **Cables rojos:** se corresponde con la conexión de 5V de energía, que suministra el pin *VU* (Vcc USB) de la placa, a los sensores.
- **Cables negros:** toma a tierra de los sensores, pin *G* de la placa.
- **Cables verdes:** *D1* y *D2* se corresponden con los pines *5* y *4* de la placa respectivamente, siendo el primero el del sensor más cercano a la salida del recinto, y el último el más interno al recinto.

Ultrasónico

En cuanto a la conexión con el sensor ultrasónico, el esquema de conexión sería como se indica en la Figura 4.31.

- **Cable rojo:** se corresponde con la conexión de *5*V de energía, que suministra el pin *VU* (Vcc USB) de la placa, al sensor.
- **Cable negro:** toma a tierra del sensor, pin *G* de la placa.
- **Cable verde:** *D2* que se corresponde con el pin *4*, que está conectado al "Echo" del sensor.
- **Cable azul:** *D1* que se corresponde con el pin *5*, que está conectado al "Trigger" del sensor.

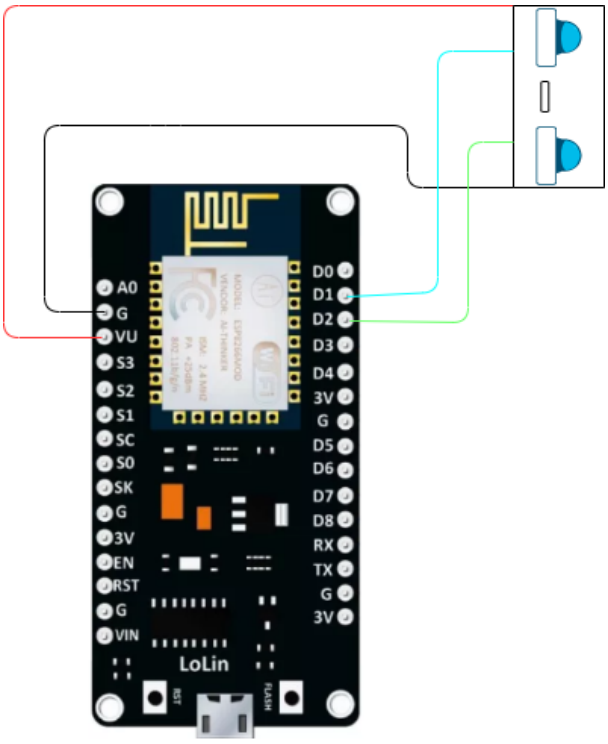


Figura 4.31: Esquema de la conexión de un sensor *HC-SR04* a la placa.

Para establecer una conexión correcta, se deben tener instaladas las librerías "PubSubClient", en este caso se ha utilizado la versión desarrollada por Nick O'Leary en su versión 2.8.0, y "Time", de Paul Stoffregen, en su versión 1.6.0.

4.2. Rendimiento

Para mejorar el rendimiento del código de detección con redes neuronales, fue necesario hacer pruebas y variar entre las tecnologías para procesamiento de imagen. Este modelo se inició entorno a una base, la cual se obtuvo a partir de un código [22] con licencia MIT License (MITL), el cuál precisa mencionar del *Copyright (c) 2020 Sai Subhakar T.*

El primer código que se realizó fue sobre el modelo de PY usando para backend la librería de OpenCV, pero poco después se probaría si un lenguaje estático como C++ funcionaría mejor. El concepto se basa en un bucle que vaya recorriendo los fotogramas asociados a una imagen, con la intención de, obtener por cada vuelta una serie de BB que posteriormente se puedan seguir.

En cada vuelta se aplica un método llamado *blobFromImage*, al que se introduce el fotograma (con *1*, *3* o *4* canales), un factor de escala que por defecto se sitúa a 1.0 (no escalado), o bien, $1/\sigma$ (para escalar los píxeles de 0 a 1) un tamaño de entrada para la FRCNN como puede ser *320×320*, *224×224*, *128×128*, etc. y finalmente, la elección

entre los *32*-bit de un float o los *8* bit de un unsigned. Una vez detectados y clasificados los elementos en BB, se aplica NMS y los BB seleccionados se vuelcan al módulo de seguimiento, definiendo que si la posición anterior está en un lado de un umbral y la nueva posición está en el lado contrario, se detecta una E/S.

Se ha probado la comparación que existe entre aplicaciones que se ejecutan sobre lenguajes estáticos y dinámicos en detecciones que exigen bastantes recursos en el dispositivo. La librería de OpenCV está disponible para múltiples lenguajes, debido a esto se escogió uno de los disponibles. Para saber con cuál se seguiría desarrollando el código, se hicieron una serie de pruebas de perfilado del mismo y otra serie de análisis estudiados con la herramienta de *Perf*.

Este modelo propuesto para el mapa de calor y la detección de E/S, necesitaba de una mejora sustancial de la ejecución sobre la RPI. Se probó a mejorar el código configurando elementos de la librería de OpenCV, pero aún es una librería en desarrollo y muchos elementos no funcionan de manera correcta, como ocurre en el caso de OpenCL. Viendo que la detección necesitaba mejorar, se decidió implementar otra alternativa como TFL (más ligero que TF), que incorpora una API junto con un interprete y ayuda a configurar elementos como el número de hilos que van a usar, cuándo invocarlo o elementos como qué FRCNN podemos establecer.

Una vez comparados los códigos con ambas librerías, se procedió a la mejora y configuración del código con mejores resultados, para obtener el mejor rendimiento posible. En total se compararon *4* códigos diferentes, que se expondrán en el Capítulo 5, para poder analizar la serie de cambios realizados. En este caso, se verá el concepto de *skip frames*. Esto ayuda a una ejecución del código en el que no se detecta cada fotograma leído, sino cada ciertos fotogramas, para no sobrecargar la detección a cambio de perder eficiencia.

4.3. Estructura del conjunto

Una vez explicadas todas las aplicaciones que componen este proyecto, se va a ilustrar cual es la estructura final del mismo (ver Figura 4.32). Todo el proyecto se compone de cinco aplicaciones, estas aplicaciones son: aplicación web, detección de ocupación mediante IA, mapa de calor, detección de ocupación mediante sensores PIR y detección de ocupación mediante sensores HC-SR4. Todas ellas se encuentran conectadas mediante MQTT por lo que es necesario tener un broker que se encargue de repartir los mensajes entre las aplicaciones.

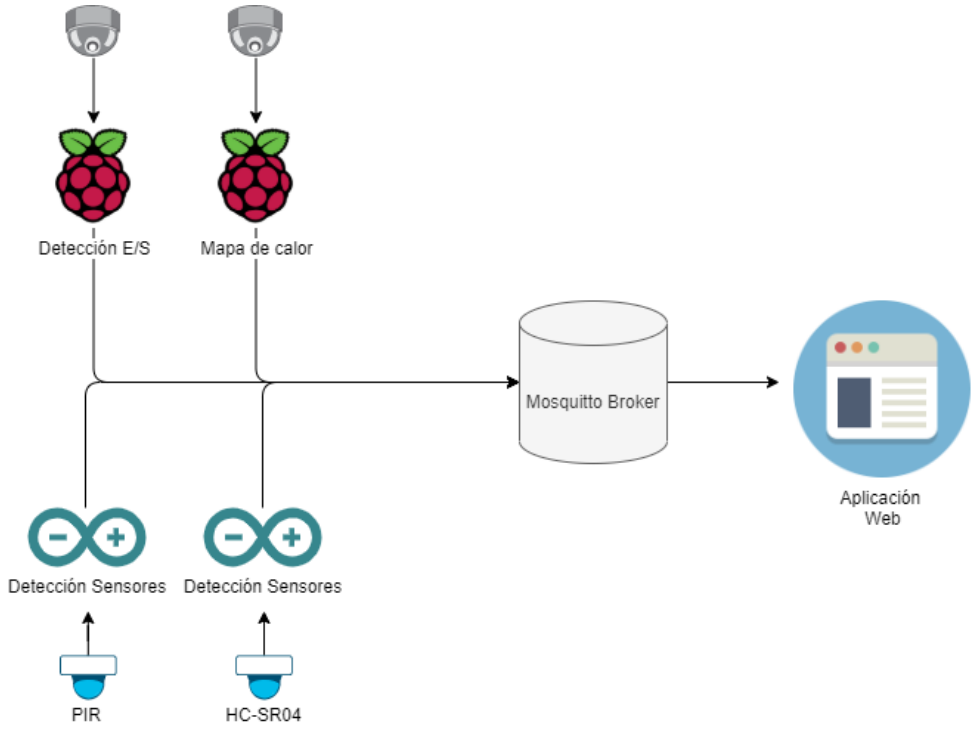


Figura 4.32: Esquema de la estructura del proyecto.

Los diferentes mensajes que se envían de los clientes a la aplicación web siguen el flujo de la Figura 4.33. Este flujo consiste en la captura constante de información hasta que uno de los clientes detecta una alteración en el escenario. En ese caso se envía el mensaje al broker, que posteriormente lo reenvía a los suscriptores. Al llegar un nuevo mensaje, este se añade a la cola de procesamiento de mensajes del suscriptor, y es almacenado en la base de datos para finalmente mostrarlos.

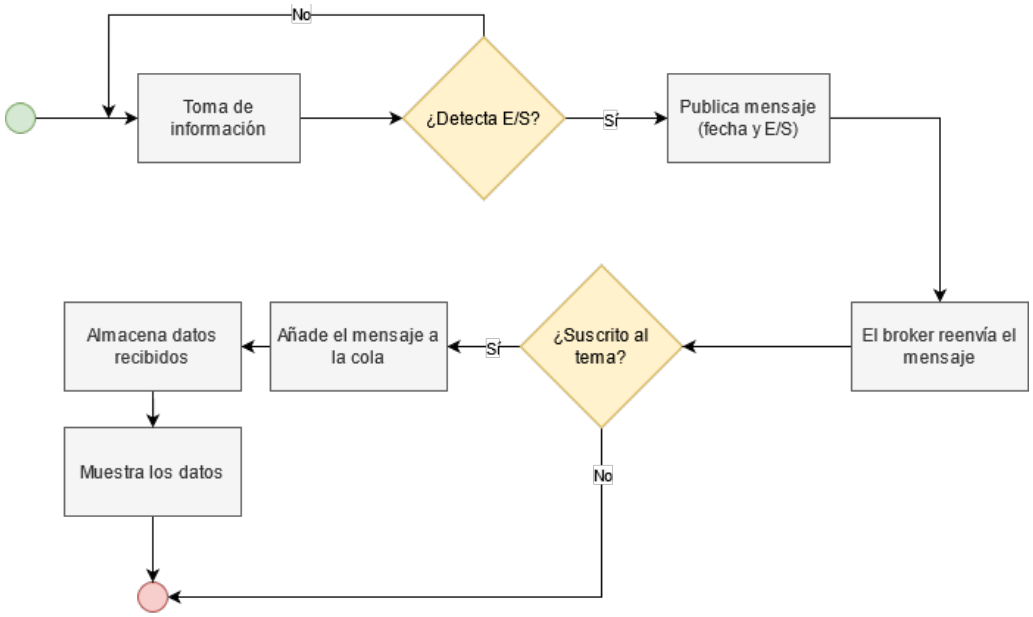


Figura 4.33: Flujo envío mensaje por eventos.

Capítulo 5

Resultados experimentales

En el capítulo anterior se ha definido el modelo base de este proyecto, a partir de él, se van a realizar una serie de pruebas que tienen como objetivo comprobar la fiabilidad y robustez del mismo.

5.1. Configuración de escenarios

Debido al gran número de configuraciones que otorga este sistema, se ha decidido configurar el siguiente escenario para ejecutar todas las pruebas (ver Figura 5.1):

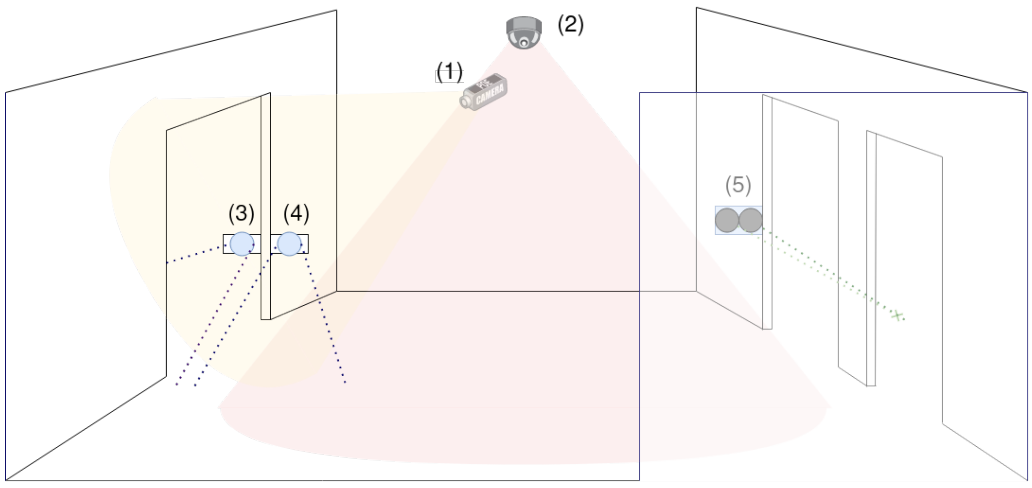


Figura 5.1: Disposición en el escenario de los componentes.

- Se ha configurado una cámara fija (ver Figura 5.1 índice (1)), situada en el interior que apunta a una puerta, por la cual van a entrar y salir las personas, y otra cámara (ver Figura 5.1 índice (2)) que apunta al interior de una sala.

- Se han dispuesto un sensor PIR HC-SR501 a un lado de la puerta y otro sensor PIR HC-SR501 al otro lado (ver Figura 5.1 índices (3) y (4)).
- Se ha ubicado un sensor HC-SR04 (ver Figura 5.1 índice (5)) dentro del recinto, de manera independiente.

5.1.1. Dispositivos Empleados

Como uno de los objetivos del proyecto era poder realizarlo en hardware de bajo coste, se han empleado los siguientes dispositivos:

Raspberry Pi 4 [23]

- Broadcom BCM2711, cuatro núcleos Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz.
- 2GB, 4GB u 8GB LPDDR4-3200 SDRAM (dependiendo del modelo).
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE Gigabit Ethernet.
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header.

NodeMCU V3 [24]

- CPU 32-bit RISC Tensilica Xtensa LX106 running at 80 MHz.
- 64 KB of SRAM.
- 512 KB of EEPROM.
- IEEE 802.11 b/g/n Wi-Fi.
- Digital I/O Pins (DIO): 16.
- Analog Input Pins (ADC): 1.
- UARTs: 1 on dedicated pins, plus a transmit-only UART can be enabled on GPIO2.
- 80 Mhz clock.

5.2. Límites de los experimentos

Previo a realizar los ensayos, se han tenido que definir una serie de limitaciones. Dado que los sensores y la cámara estarán apuntando a una puerta, la longitud de esta no ha de ser mayor de tres metros, ya que, el sensor ultrasónico puede recoger valores erróneos a

partir de esa distancia. Cabe destacar, que el acceso no ha de tener una puerta explícitamente, pero lo que se tiene que dar para ser considerado como acceso, es que las personas al pasar por ahí puedan entrar o salir. También se ha tenido que realizar la definición de lo que es una persona, ya que esta debía ser detectada independientemente de su raza y medidas antropométricas [25].

Por último, se ha de mencionar que la forma de tomar las pruebas será con vídeos establecidos previamente, y teniendo situado cada componente del sistema propuesto a una distancia geográfica considerable. La razón de esta limitación se ha generado por la actual crisis sanitaria, en la cual hacer pruebas en situaciones reales, complica el estudio de una prueba de aforo habitual. Todo esto hace que la prueba del modelo concreto se haga a través de una *VPN*[1], y la causa de no poder tomar datos en un escenario real.

Otro punto a tener en cuenta, es que para el correcto funcionamiento del sensor ultrasónico, sólo puede usarse en entradas unidireccionales dobles (entradas y salidas separadas). Por lo que se incluirá en pruebas globales de rendimiento sobre el modelo completo, suponiendo la posición de este, de manera independiente.

5.3. Pruebas realizadas y resultados

5.3.1. Aplicación Web

Para las pruebas de la web, además de realizar de forma periódica el testeo manual de implementaciones o características añadidas, acompañado de un período de bug fixing, se han definido y llevado a cabo las dos pruebas expuestas en la tabla de pruebas Web (ver Tabla 5.1).

ID	**Descripción**	**Repeticiones**
WEB-01	Representación Heatmap	-
WEB-02	Comunicación BBDD	-

Tabla 5.1: Pruebas Web realizadas.

WEB-01

En esta primera prueba, referente a la parte de heatmap, se ha modificado el tamaño de cada celda de la matriz que se superpone a la imagen para representarlo. Es decir, para un mismo vídeo, de resolución 352×640 píxeles, se han usado matrices donde las celdas son de tamaño un píxel, en este caso la matriz se corresponde con la resolución del vídeo, cinco, diez y veinte píxeles.

[1] Red privada virtual

Los resultados se han obtenido monitorizando el inicio y finalización de la impresión por pantalla, y se han recogido en una tabla, para posteriormente calcular el tiempo medio total por cada tamaño y representarlo en una gráfica (ver Figura 5.2).

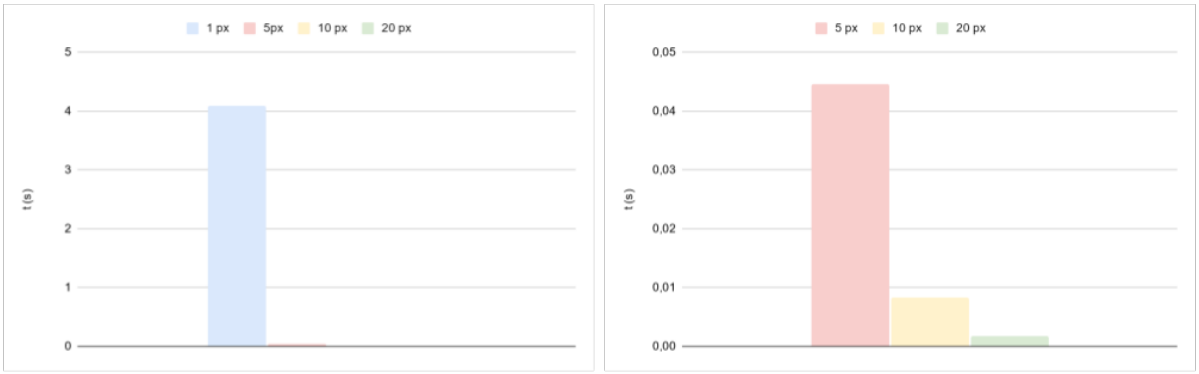


Figura 5.2: Tiempo de representación por tamaño de celda.

Además, los distintos tamaños se pueden ligar a cuatro resultados visuales reflejados en la Figura 5.3.

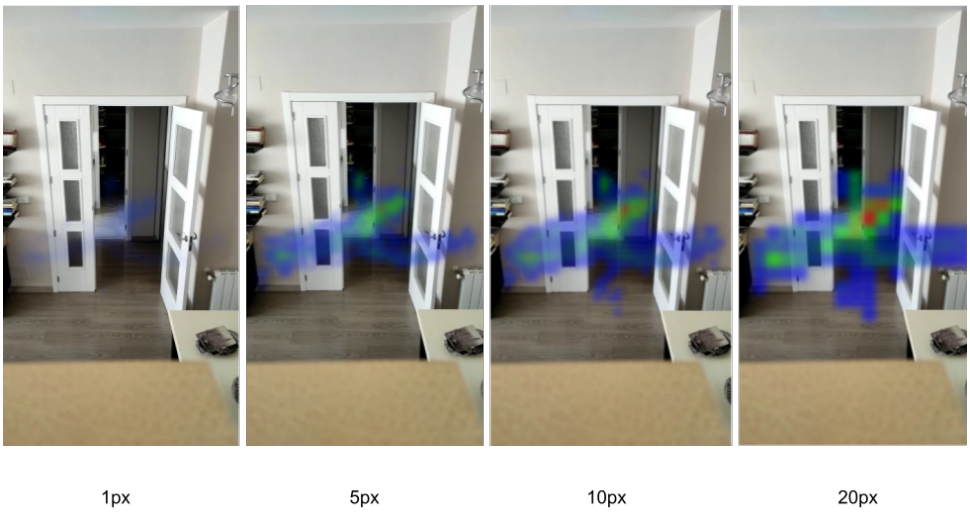


Figura 5.3: Mapas de calor según tamaño de celda.

Con esta prueba, se descartaron inmediatamente, tanto el tamaño de un píxel, porque el tiempo de representación es insostenible, como el de veinte píxeles, ya que no se consigue una representación lograda y además hay una pérdida de información demasiado grande. Finalmente, se concluyó que el mejor tamaño sería el de *10* píxeles puesto que, el de *5*, aún no llegando en este caso a sobrepasar los *45* milisegundos, es posible que no deje suficiente margen para representar vídeos con mayor resolución.

WEB-02

Para esta parte, se realizaron pruebas de pérdida de conexión con la base de datos en diferentes situaciones como cargar estadísticas, hacer login o persistir datos. Todas ellas demostraron la correcta gestión de errores y excepciones. Mostrando un mensaje de error al usuario adecuado para cada situación.

5.3.2. IoT

En esta sección se va a hablar sobre los resultados obtenidos en las diferentes pruebas realizadas para comprobar la robustez del intercambio de mensajes entre aplicaciones. Todas las pruebas se realizarán con el mismo broker y en ellas se introducirán diferentes configuraciones. Para obtener los resultados se van a utilizar dos aplicaciones:

- **Wireshark** que permite ver los datagramas enviados en una red y su contenido.
- **MQTT Explorer** que permite conectarse al broker actuando como un suscriptor de todos los temas, pudiendo monitorizar los mensajes recibidos.

Además se ha fijado el flag de MQTT que establece la sesión en false, de esta forma, en caso de pérdida en la red y reconexión, el broker envía el último mensaje de cada tema a los diferentes suscriptores. También es importante indicar que todas las pruebas se han realizado para los diferentes tipos de QoS explicados, sin embargo, una de las aplicaciones no tiene la posibilidad de cambiar de QoS[2], por lo que no se han tenido en cuenta sus datos.

Las pruebas que se van a realizar están contenidas en la Tabla 5.2.

ID	Descripción	Repeticiones
IOT-01	Configuración QoS	-
IOT-02	Retardo de los Mensajes	-
IOT-03	Conexión y Reconexión	-
IOT-04	Envío con alteraciones en la red	-

Tabla 5.2: Pruebas IoT realizadas.

IOT-01

En estas pruebas se van a comprobar las diferencias entre los niveles de servicio que MQTT ofrece. Para ello se va a cambiar la QoS y a reproducir el mismo vídeo. De esta forma, se enviarán los mismos datos y se podrán observar las diferencias.

[2] https://www.hivemq.com/blog/mqtt-client-library-encyclopedia-arduino-pubsubclient/

QoS 0 El número de mensajes enviados al establecer esta QoS es de uno (ver Figura 5.4). Ya que, como se ha explicado anteriormente, en este caso no se comprueba si el mensaje llega o no. Es por esto que, en caso de pérdida de la conexión con el broker, el mensaje no llega.

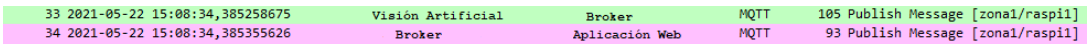


Figura 5.4: Ejemplo envío mensajes con QoS 0.

QoS 1 En este caso el número de mensajes que se envían es de dos (ver Figura 5.5). En caso de que el suscriptor no se encuentre activo, este guardará el mensaje hasta que lo pueda enviar.

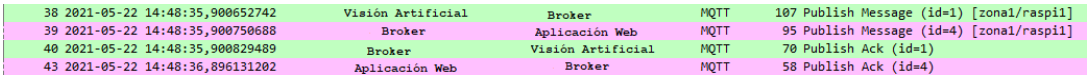


Figura 5.5: Ejemplo envío mensajes con QoS 1.

QoS 2 En esta ocasión el número de mensajes que se necesitan, como se ha visto anteriormente, es de cuatro (ver Figura 5.6). Esto hace que el mensaje se envíe en caso de que haya una pequeña pérdida en la red que no deje llegar al ACK de confirmación

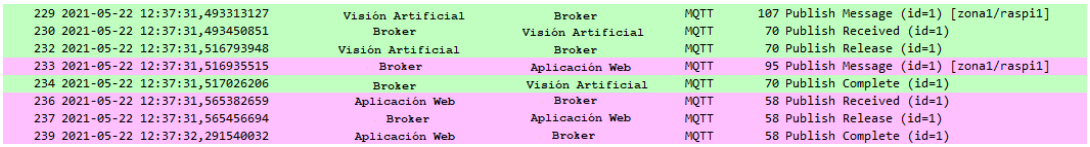
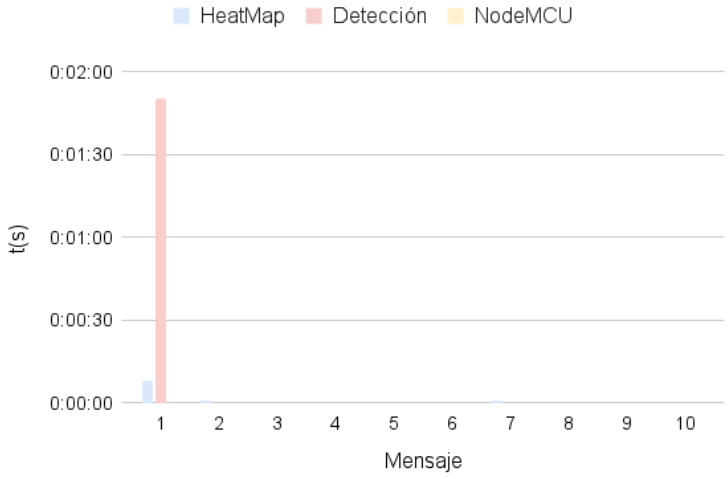


Figura 5.6: Ejemplo envío mensajes con QoS 2.

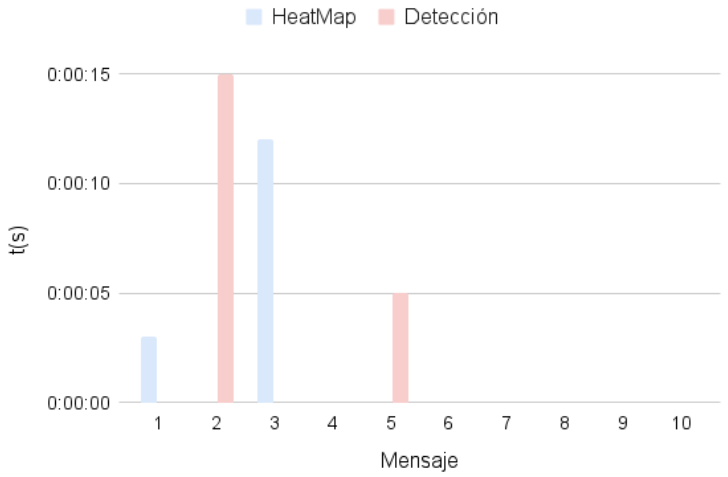
IOT-02

Con esta prueba, se van a obtener los diferentes tiempos de los mensajes desde que se envían por las diferentes aplicaciones al broker, hasta que este los reenvía. Para ello, se va a comprobar el momento exacto en el que el mensaje es enviado y el instante en el que se recibe. Debido a que los tiempos menores a un segundo apenas son apreciables, se han tenido en cuenta aquellos mayores.

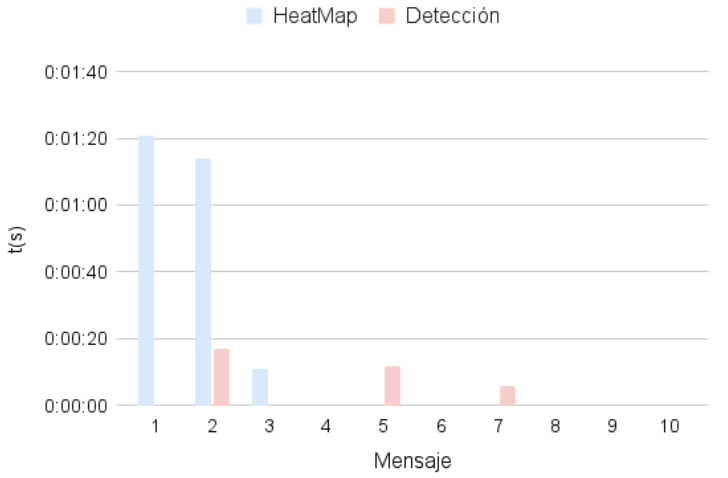
Con los datos obtenidos en la prueba (ver Figura 5.7), se puede ver como los tiempos entre el envío y la recepción de los mensajes con QoS 0 es superior a las otras dos. Además, hay que tener en cuenta que los mensajes 4 y 6 no han llegado con esta calidad. Tras esto, observamos que los tiempos de la QoS 2 son mayores que los de QoS 1 por lo que, para las aplicaciones que lo soportan el idóneo es QoS 1.



(a)



(b)



(c)

Figura 5.7: Retardo de mensajes en: (a) QoS 0. (b) QoS 1. (c) QoS 2.

IOT-03

Con estas pruebas, se va a comprobar cuanto tiempo tarda en reconectarse. Para ello, se mide el momento desde que se envía el Connect Command (ver Figura 5.8) hasta que se recibe el ACK. Para obtener estos tiempos se ha utilizado Wireshark. Además, se ha mirado cuántos mensajes se pierden con estas reconexiones. Esto se ha realizado para todas las calidades de servicio, excepto para el caso de los *NodeMCU* como anteriormente se ha indicado.

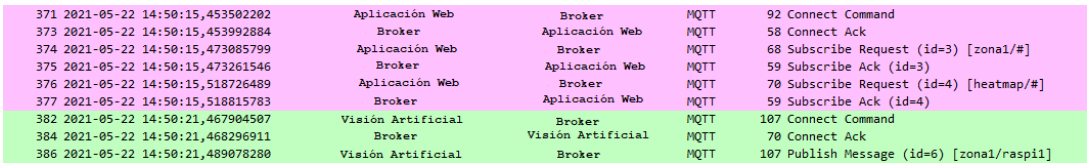


Figura 5.8: Ejemplo de conexión.

Tras realizar varias pruebas que afectasen a la red, se han obtenido estos datos, (ver Tabla 5.3 y Tabla 5.4) con los cuales se puede tener una idea del tiempo que tarda en reconectarse cada una de las aplicaciones, y el número de paquetes que se han perdido a lo largo de la prueba. Cabe destacar que la cantidad de paquetes que se han perdido para la QoS 0 es muy superior al resto, y que los tiempos de conexión y reconexión apenas se ven afectados por el tipo de QoS utilizada.

	Conexión QoS 0		Conexión QoS 1		Conexión QoS 2	
	Tiempo(ms)	Paquetes perdidos	Tiempo (ms)	Paquetes perdidos	Tiempo(ms)	Paquetes perdidos
Detección	34	0	39	0	46	0
Heatmap	33	0	37	0	42	0
NODE MCU	22	0	No hay datos		No hay datos	
WEB	492	0	527	0	398	0

Tabla 5.3: Pruebas de conexión.

	Reconexión QoS 0		Reconexión QoS 1		Reconexión QoS 2	
	Tiempo(ms)	Paquetes perdidos	Tiempo(ms)	Paquetes perdidos	Tiempo(ms)	Paquetes perdidos
Detección	26	7	39	0	26	0
Heatmap	33	6	38	2	25	0
NODE MCU	54	6	No hay datos		No hay datos	
WEB	365	0	651	0	363	0

Tabla 5.4: Pruebas de reconexión.

IOT-04

Para esta prueba se han enviado durante un período de tiempo la misma cantidad de mensajes. De esta forma, se puede obtener el porcentaje de llegada simulando diferentes errores en la red. Para ello, se ha quitado la conexión durante el envío y se ha apagado el broker, para obtener en ambos casos los paquetes perdidos. Esto se ha realizado para todos los tipos de QoS.

Con todos los datos anteriores (ver Figura 5.9 y Figura 5.10), se ve que tras un periodo de tiempo de envío de mensajes, la QoS 0 da una media de *6,33* mensajes perdidos, por lo que llegan un *64.81 %* de los mensajes, y un tiempo medio de *10* segundos entre el envío y la recepción del mensaje. La QoS 1 ofrece tan solo 1 paquete perdido de media, un porcentaje de llegada de un *88.89 %* y un tiempo de *7* segundos. Finalmente la QoS 2 ofrece un *100 %* de mensajes correctos, aunque en este caso el tiempo medio crece hasta *13* segundos.

Teniendo presente que los tiempos cuentan la reconexión y los tiempos de los mensajes, se puede concluir que la QoS recomendada es la QoS 1 porque es la que ofrece un mejor rendimiento en relación al tiempo de envío y recepción.

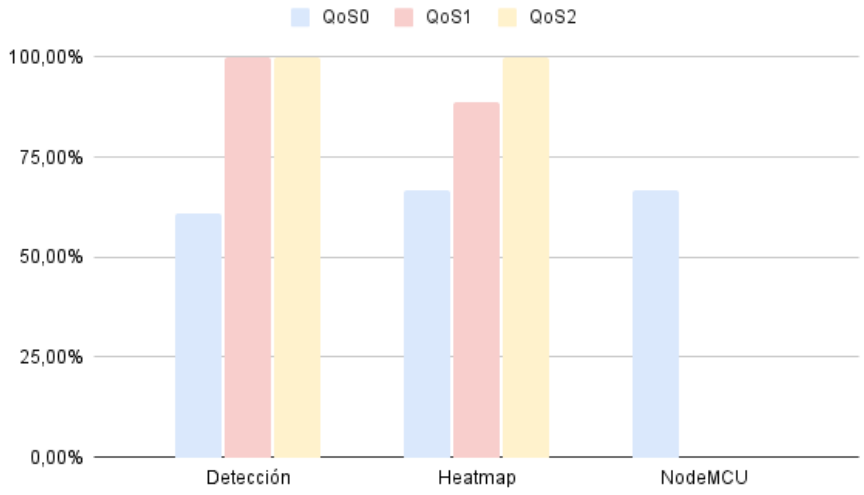


Figura 5.9: Porcentaje de llegada de datos.

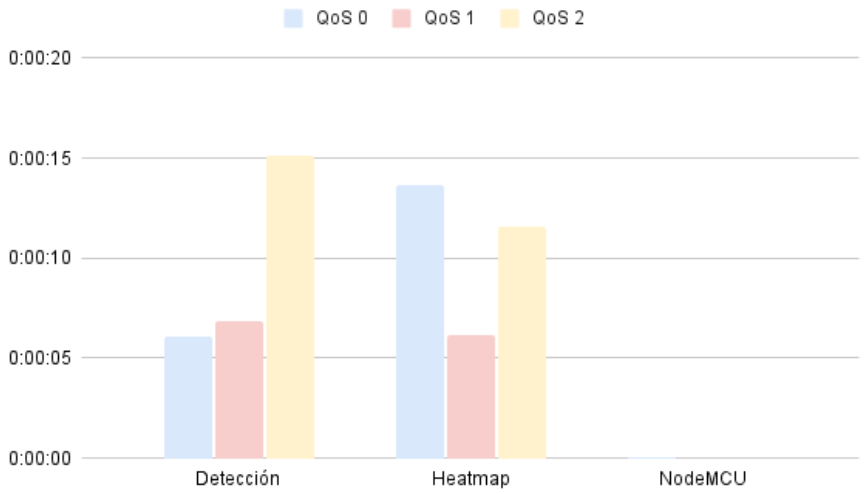


Figura 5.10: Tiempo medio (s) entre envío y recepción.

Conclusión

En función de los datos obtenidos en las pruebas realizadas, se puede decir que la QoS más apropiada para esta aplicación es QoS 1, ya que, es con la que se obtiene un mayor porcentaje de llegada en caso de sufrir alguna caída, y la que mejor tiempo entre el envío y la recepción ofrece. La QoS 2 ofrece según las pruebas un 100 %, pero los tiempos son considerablemente mayores. Por el contrario, QoS 0 no ofrece una gran fiabilidad en caso de que haya algún problema en la red, por lo que se ha rechazado esta opción.

5.3.3. Reconocimiento y clasificación de imágenes

Se han realizado una serie de pruebas referentes a un estudio de rendimiento, junto con sus consiguientes pruebas de eficiencia sobre la detección de imagen y posterior control de aforo (ver Tabla 5.5).

Módulo	**ID**	**Descripción**	**Repeticiones**
Rendimiento	*VA-01*	Estadísticas	48
	VA-02	Variación de los parámetros	24
	VA-03	Hilos utilizados	32
	VA-04	Perfilado de los códigos	42
	VA-05	Detección correcta	16
Pruebas de detección	*VA-06*	Detección entrada y salida de personas	-
	VA-07	Mapa de calor	12
	VA-08	Contraluz	-

Tabla 5.5: Pruebas VA realizadas.

Códigos empleados

Los códigos empleados para hacer las diferentes pruebas, reciben los siguientes nombres:

- *Python*: Realizado enteramente en Python, cuenta con un código que emplea un backend en OpenCV.
- *COpenCV*: Código escrito en C++, a partir del empleado en Python, que de igual manera aplica ese mismo backend.
- *TFL*: Programa el cual utiliza la TensorFlow-Lite API para el módulo de detección y clasificación de imágenes.
- *TFLSKIP*: Similar al código de TFL, con la diferencia de que en este se aplica skipframes.

Previo a la realización de pruebas, cabe destacar la importancia de los vídeos aplicados sobre las pruebas, como se muestra en la Tabla 5.6.

ID	Descripción breve	Resolución
VID1	Abundancia de individuos cruzando una puerta giratoria de escasa duración.	1280x720
VID2	Abundancia de individuos cruzando una serie de puertas automáticas, donde se hace zoom al final del vídeo y así se pueden obtener resultados con diferentes tamaños de objetos.	1280x720
VID3	Flujo normal de individuos, interior de un hogar en el que como máximo se llegan a cruzar hasta cuatro personas a la vez. Utilidad tanto para control de E/S, como para mapa de calor.	640x352
VID4	Tránsito de una serie de personas por la calle, vista perpendicular al suelo y flujo normal de individuos.	480x480
VID5	Detección sobre persona *1* con buena iluminación con y sin mascarilla, añadiendo una serie de movimientos sobre el mismo sitio.	3840x2160 1280x720
VID6	Detección sobre persona *1* con mala iluminación con y sin mascarilla, añadiendo una serie de movimientos sobre el mismo sitio.	3840x2160 1280x720
VID7	Detección sobre persona *2* con buena ilumniación con y sin mascarilla, añadiendo una serie de movimientos sobre el mismo sitio.	3840x2160 1280x720
VID8	Flujo de E/S con buena iluminación, para situación de pocos individuos.	3840x2160 1280x720
VID9	Flujo de E/S con mala iluminación, para situación de pocos individuos.	3840x2160 1280x720
VID10	Flujo de comportamiento, con pocos individuos, para mapa de calor con buena iluminación.	3840x2160 1280x720
VID11	Flujo de comportamiento, con pocos individuos, para mapa de calor con mala iluminación.	3840x2160 1280x720
VID12	Elevada exposición de luz dónde un individuo realiza una serie de movimientos.	3840x2160 1280x720

Tabla 5.6: Descripción e identificadores de los vídeos aplicados sobre las pruebas.

Rendimiento

VA-01

Se va a tratar el conjunto de todos los códigos con la finalidad de ver cuál ejecuta los vídeos propuestos con mayor celeridad, y cuál aprovecha los recursos del dispositivo al máximo. Para poder realizar las pruebas, se escogieron una serie de vídeos con gran afluencia de personas, como son *VID1*, *VID3* y *VID4*, para estimular la RPI y conseguir un gran empeño de la misma.

En primer lugar, se ejecutaron los dos códigos de partida, para saber con cual se debía seguir continuando el desarrollo. Como se puede ver en la Figura 5.11, de los códigos seleccionados donde se estudiaba la ejecución con RRNN tipo SSD, se obtuvo mejor rendimiento para *COpenCV*.

A continuación, estos códigos serán ejecutados utilizando clasificaciones mediante RRNN del tipo SSD Lite, esto se debe a la comparación entre códigos con una red neuronal similar, siendo TFL empleada con modelos exclusivamente ligeros.

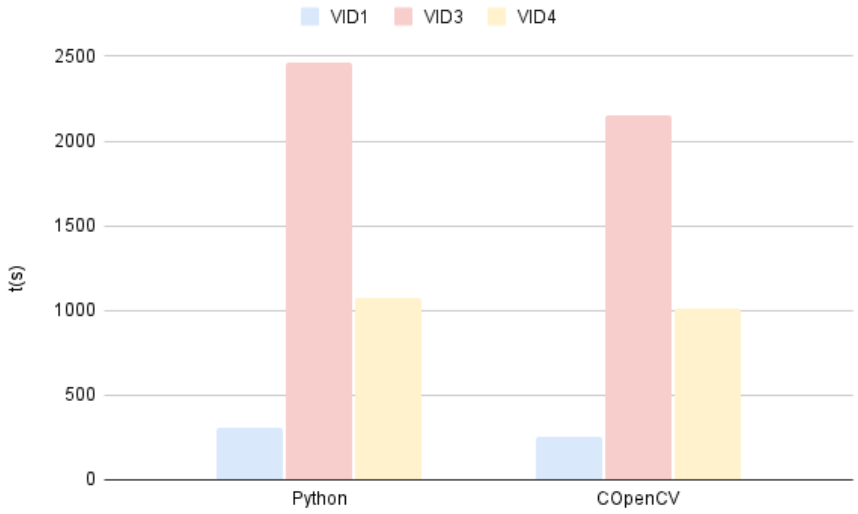


Figura 5.11: Estadísticas de ejecución primeros códigos.

Se ha de recalcar que en la Figura 5.12, las ejecuciones realizadas sobre los vídeos son relacionadas con el tiempo que tardan en completarse. Analizando la Tabla 5.7, junto con la gráfica, se puede concluir que el aprovechamiento, en cuestión de núcleos empleados, no implica un menor tiempo. Y que al emplear otro tipo de RRNN más ligeras, aunque menos precisas, se obtiene un menor tiempo de ejecución en el código de *COpenCV**[3].

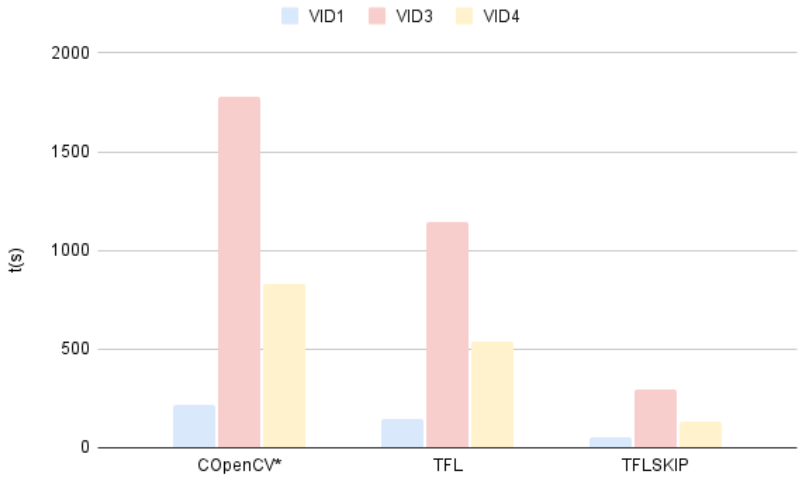


Figura 5.12: Estadísticas de ejecución de los códigos finales.

En el caso de *TFL* y de *TFLSKIP*, se debe al procesamiento de imágenes de manera intermitente, con lo que llega a variar de manera proporcional a la cantidad de fotogramas sorteados.

[3]Código COpenCV aplicando RRNN tipo SSD Lite.

Código	Media de núcleos empleados
Python	3,19
COpenCV	3,45
*COpenCV**	3,00
TFL	2,29
TFLSKIP	1,82

Tabla 5.7: Media de núcleos lógicos utilizados por cada programa ejecutado.

VA-02

Para obtener el mejor resultado posible, es necesario que todos los parámetros estén bien ajustados. Se analizarán los cambios en los parámetros estudiando: la entrada de la red neural, el umbral para la función IOU y el valor de confianza de la clasificación de objetos.

A la hora de tomar las mediciones, se obviarán valores sobre el umbral de IOU como el *0* o el *1*. La razón de esta decisión se debe a que ambos representan tanto la unión nula de dos BB, como la totalidad de la unión. La entrada de la red neuronal tomará valores comunes a las utilizadas para el entrenamiento [17].

Estudiando la Tabla 5.8, se representan los valores asociados al acierto en relación a la medida real. Una vez realizada la prueba, se obtuvo que, sólo los valores con entrada *320×320* son los que han detectado algún objeto, para entradas tales como *128×128*, *224×224*, *480×480* o *640×640*, no se logró ninguna detección. Para valores de confianza de *67 %*, el modelo detecta mejor a las personas, ya que se trata de un modelo de SSD Lite que no estima una precisión alta de clasificación.

Clasificaciones por vídeo (%)	IoU Threshold	Confidence Value		
		0,65	0,67	0,7
VID3	0,3	109,98	102,91	45,94
	0,6	118,04	102,78	45,94
VID5	0,3	100,00	100,00	10,62
	0,6	100,00	100,00	10,62
VID6	0,3	100,00	100,00	0,00
	0,6	100,00	100,00	0,00
VID7	0,3	99,65	98,86	45,94
	0,6	99,65	98,86	45,94

Tabla 5.8: Porcentajes de aciertos sobre clasificación de objetos.

Se han observado resultados superiores a *100 %*, como anteriormente se ha explicado, debido a que ciertas detecciones fueron determinadas como falsos positivos y de ahí, el exceso de porcentaje obtenido.

Finalizando este estudio, se acota el valor entre *0,3* y *0,6* para el umbral de IOU, en el que se ha optado por utilizar un valor donde varias BB puedan estar más próximas entre sí, en este caso *0,6*.

VA-03

Otro de los ajustes a realizar sobre *TFL* consiste en variar el número total de hilos utilizados, para ver cómo afecta al rendimiento del procesamiento de imagen. Para ello, se escogen una serie vídeos con alta densidad de BB, con el propósito de ver el comportamiento en los casos de más necesidad.

El conjunto de hilos seleccionados se establecerá en función de los núcleos lógicos disponibles en la RPI, teniendo esta, un único núcleo lógico asociado a cada uno de los cuatro núcleos físicos. Elevando la cantidad de hilos lanzados al sistema operativo, se pretende sobrecargar la cola de tareas frente a otras ajenas al programa del modelo.

Teniendo en cuenta la Figura 5.13 y la Figura 5.14, se llega a la conclusión de que los valores con mejores resultados son los programas ejecutados con *8* y *16* hilos, en relación al uso real de hilos y tiempo de ejecución (siendo el menor, el preferible). Finalmente, se tomará el valor de *16* hilos debido a que utiliza *1,76* núcleos empleados de media y es, dentro de las tomas realizadas, el modelo que mejores tiempos hizo (exceptuando una de las medidas que fue superado por el modelo de *8* hilos).

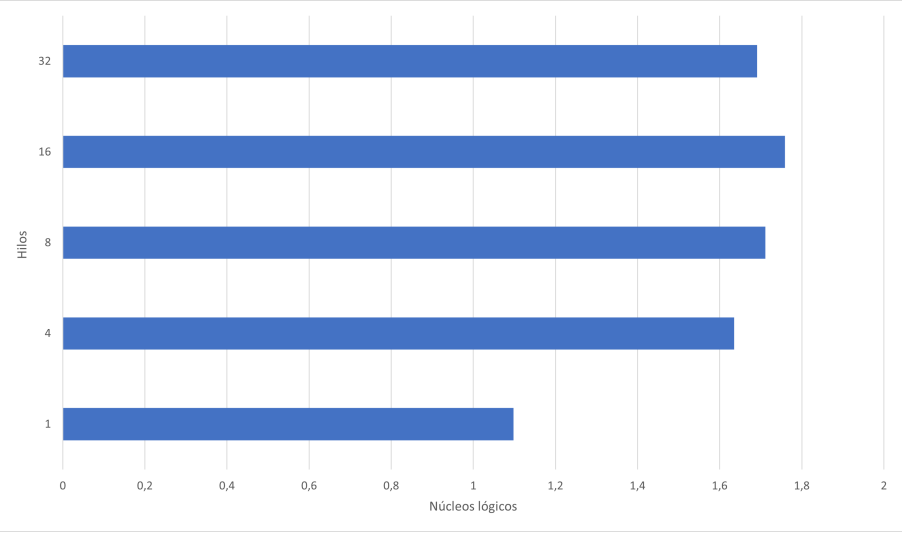


Figura 5.13: Núcleos lógicos empleados en relación a los hilos seleccionados desde la API.

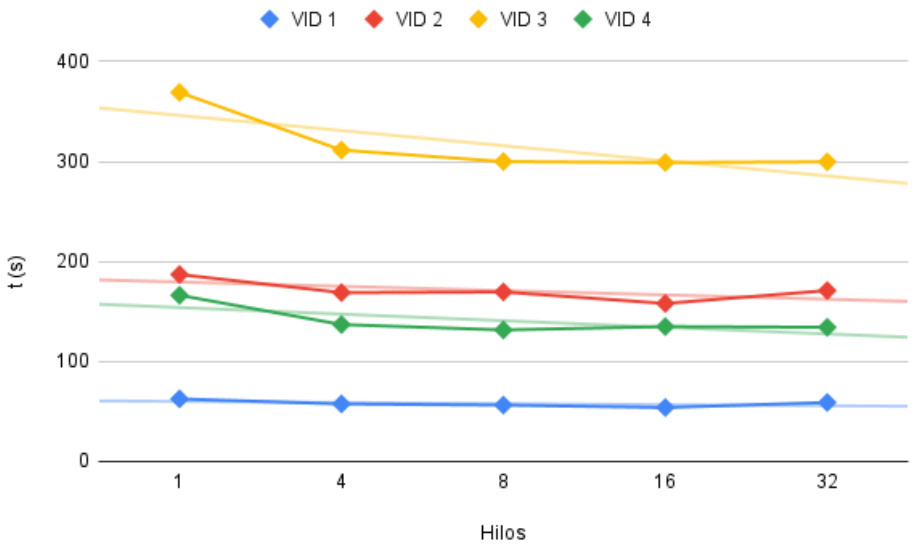


Figura 5.14: Tiempo transcurrido por vídeo (s).

VA-04

Esta prueba trata de obtener un perfilado del código ejecutado para las diferentes tecnologías y ver la sobrecarga que se puede producir. Para poder realizar la prueba se escogieron una serie de vídeos que contuviesen elevadas cantidades de clasificaciones de imágenes y al igual que en pruebas previamente realizadas, se escogen los que más número de individuos por fotograma sea posible, o bien, gran afluencia en el mismo instante, en este caso *VID1*, *VID2*, *VID3* y *VID4*.

Como se puede observar en la Figura 5.15, cada ejecución de código está representada por las funciones con más sobrecargas ordenadas por porcentaje descendente. Las ejecuciones realizadas reflejan la necesidad de adecuar el consumo de la primera función sobre *COpenCV*, a razón del consumo excesivo (*88,87 %* de media), sobre funciones de detección y clasificación en redes neuronales, siendo distintas entre los frameworks siguiendo el mismo propósito.

Para el código *COpenCV** podemos ver que, gracias haber aplicado una clasificación de tipo SSD Lite entrada para frameworks como TFL, mejora el comportamiento sobre la carga de trabajo llegando a un máximo de *59,06 %* de sobrecarga.

En el caso de *TFL* y *TFLSKIP* se ha estudiado la cercana similitud en cuanto a la baja sobrecarga de ambos. Por lo que, a partir de este punto se procederá a continuar las pruebas sin *COpenCV* teniendo en cuenta ambas RRNN.

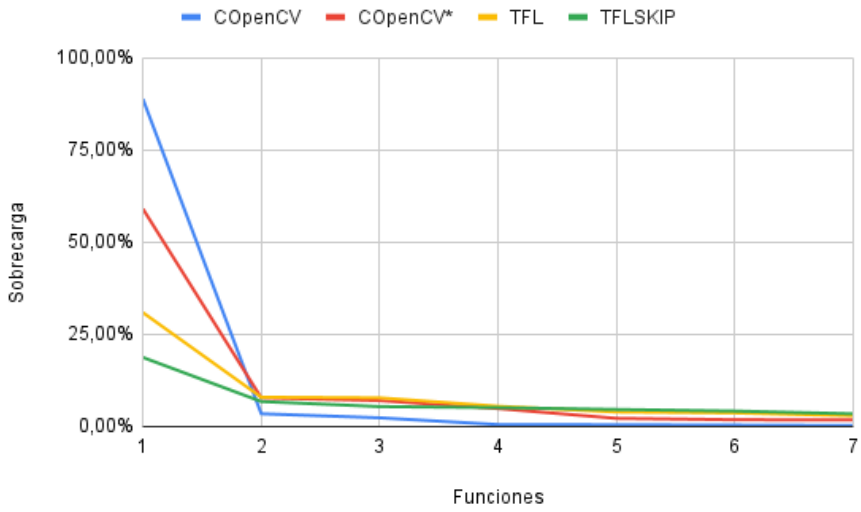


Figura 5.15: Perfilado sobre los códigos.

VA-05

Para comprobar la eficiencia frente a la ejecución del vídeo sin detección, se van a definir primero las detecciones que se producen para distintos códigos, variando el total de skipframes utilizados. Más adelante, se documentará el tiempo que tarda en ejecutarlo.

Esta prueba está pensada para comprobar si, de los códigos seleccionados, concuerda alguno con el flujo normal del vídeo. La serie de repeticiones asociadas a la prueba se efectuarán a partir de vídeos destinados a la eficiencia, como son *VID3*, *VID5*, *VID6* y *VID7*.

En referencia a la Tabla 5.9, se puede estudiar cómo el vídeo más próximo a la detección real es el de *TFL*, esto, en gran parte, se debe a que es el único que cíclicamente toma datos. Haciendo una agregación con la Tabla 5.10, se aprecia una gran lentitud del código *TFL* (837,93 % más lento) frente a la ejecución real y es por esto que su media de FPS es de aproximadamente *3,29*.

Código	Detecciones (%)
TFL	102,05
TFLSKIP 5	16,34
TFLSKIP 14	16,32
TFLSKIP 20	4,03

Tabla 5.9: Detecciones totales durante una ejecución completa.

Comparando el código con *5* y *14* skipframes, se puede llegar a la conclusión de que la diferencia no es significativa en detecciones ni en tiempo de ejecución. En el

caso de establecer los skipframes a *20*, se puede apreciar como es el más cercano al valor real de FPS, pero la pérdida de detecciones es notoria (*95,97 %* de pérdidas).

Código	Exceso de tiempo (%)	Media FPS
TFL	837,93	3,29
TFLSKIP 5	243,42	9,29
TFLSKIP 14	244,41	9,36
TFLSKIP 20	129,51	15,18
Sin código	0	30

Tabla 5.10: Tiempo total de ejecución y media de FPS.

Pruebas de detección

VA-06

Control de detección E/S con luz y sin luz, con un vídeo que trate un flujo real de aforo, de esta manera se comprueba la eficacia con cierta cantidad de personas, variando la iluminación. Para esta prueba se expusieron una serie de vídeos sin compresión, junto con una serie de vídeos comprimidos. Estos vídeos son *VID3*, *VID8* y *VID9*, donde estos dos últimos se aplicarán en su formato comprimido y el original.

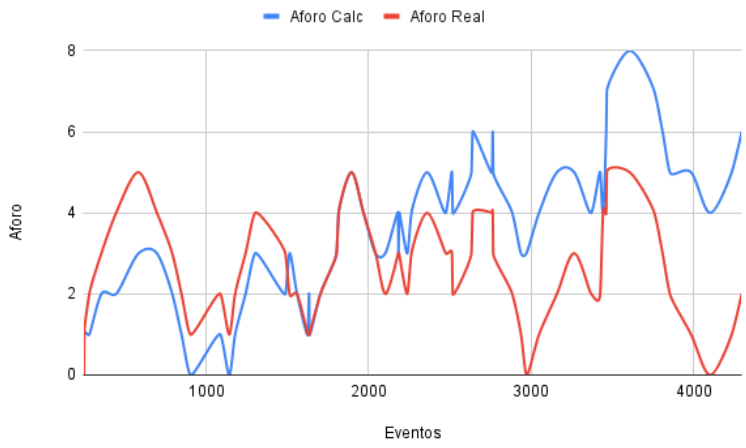


Figura 5.16: Captura E/S sobre un flujo de tráfico normal.

Tras la realización de las pruebas, se ha estudiado la similitud con datos basados en una cuenta centrada en la observación (la cual será llamada *Real*). Observando las gráficas de las Figuras 5.16, 5.17 y 5.18, se puede analizar cómo ajustando los parámetros, se obtendrán datos semejantes.

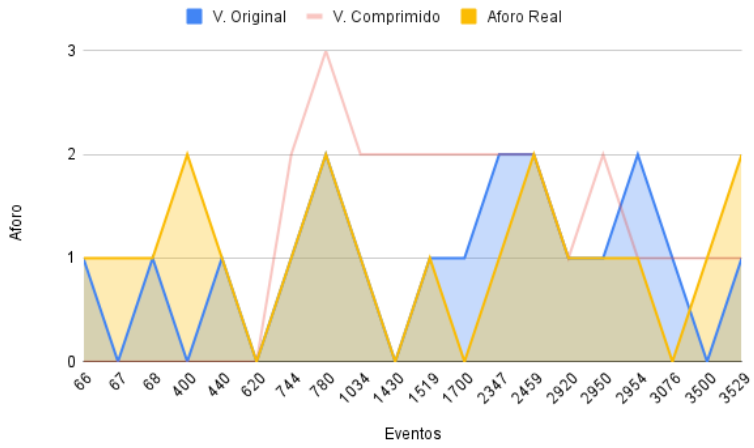


Figura 5.17: Captura E/S sobre un vídeo con luz alta.

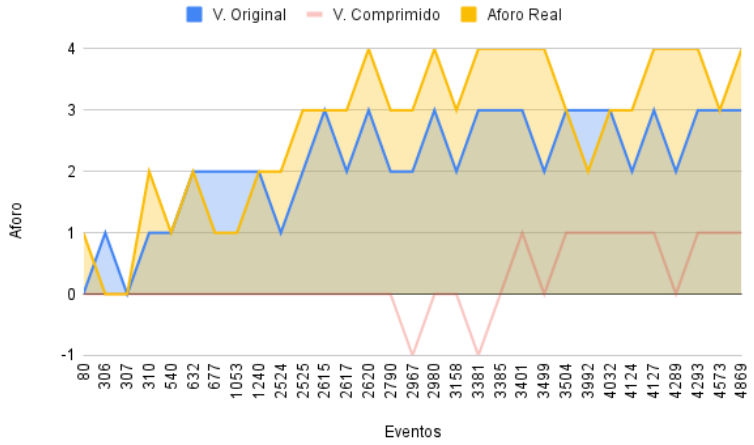


Figura 5.18: Captura E/S sobre un vídeo con luz baja.

Para saber el grado de afinidad sobre el dato capturado por el código del modelo, se aplicará el coeficiente de correlación de Pearson. La razón del uso de esta correlación, se basa en el estudio de similitud entre dos variables [26], en este caso los datos tomados, frente a su medición real.

De esta forma, se sabrá en qué medida se asemejan con valores inversamente similares (*-1*), sin asociación (*0*) y directamente semejantes (*1*). Como se puede ver en la Tabla 5.11, en el vídeo utilizado para el flujo normal de E/S, la correlación que hay entre los datos es de aproximadamente la mitad. Esto se debe a que al producir confusión sobre la medición, esta falla formando datos erróneos. A pesar de esto, siendo de los vídeos comprimidos, es el que mejores mediciones proporciona.

	Vídeo comprimido	Vídeo sin comprimir
VID3	0,535	-
VID8	0,085	0,355
VID9	0,193	0,759

Tabla 5.11: Coeficiente de correlación de Pearson sobre las pruebas.

Para el resto de vídeos de prueba (comprimidos), la diferencia entre la lectura obtenida y la esperada fue algo elevada. Por esto, sería conveniente realizar más pruebas, reajustando los parámetros y viendo el comportamiento con vídeos en alta calidad (de los que no se dispone).

VA-07

Mapa de calor en diferentes iluminaciones y para un escenario con pocas personas, empleando vídeos tales como *VID3*, *VID10* y *VID11*. Posteriormente se estudiará un escenario con mayor flujo de datos, para ver un mapa de calor con suficiente información. La idea principal que se quiere reflejar en esta prueba, se fundamenta en la recogida de datos para diferentes marcas de tiempo (medidas en fotogramas transcurridos).

El propósito de tomar datos con frecuencia variada pretende estudiar cómo representar un mapa con datos suficientes, a partir del original, siendo a su vez ligero para el alojamiento de la matriz resultante.

Observando la Tabla 5.12, podemos apreciar la pérdida de información a consecuencia del filtrado. Cada celda contiene un entero con un valor máximo de *2.147.483.647*. Teniendo en cuenta el máximo valor de cada celda y la pérdida asociada a cada filtro, se pueden establecer una serie de pruebas de resistencia y aplicar filtros considerando la falta de información.

	Filtro	
	10 fotogramas	20 fotogramas
VID3	10,05 %	5,18 %
VID10	9,97 %	4,92 %
VID11	9,96 %	5,08 %

Tabla 5.12: Datos tomados en relación a la muestra sin filtro.

Los vídeos aplicados para este proyecto pueden llegar a contener en torno a *8.294.400* píxeles. Gracias a saber el tamaño ocupado por cada celda (*4* bytes), se puede estimar un máximo (*31,64* MB) para la matriz que debería ser almacenada por la RPI en cada ejecución.

Por último, si se ocupara cada celda con el máximo del tipo *long long int*, se almacenarían un total de *63,28* MB.

VA-08

Esta prueba trata de ver la repercusión que produce la sobreexposición de luz durante toda la ejecución del vídeo. Para ello, se tomará un vídeo (ver Figura 5.19) de referencia donde se captura una gran cantidad de luz (*VID12*).

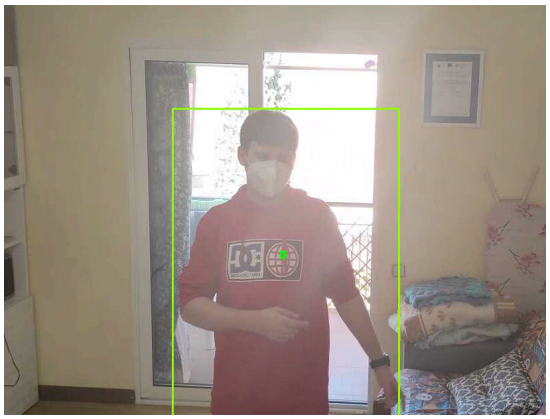


Figura 5.19: Detección de un individuo a alta exposición de luz.

Una vez realizada la prueba, como resultado se obtuvo un *95,67 %* de acierto, en base a la detección continua de una persona.

Para concluir esta serie de pruebas aplicadas al control de aforo empleando CV, se tiene en cuenta los valores con mejor resultado obtenido establecen *16* hilos empleados, un umbral de IOU de *0,6*, una entrada de la red neuronal aplicada de *320x320*, un valor de confianza de *0,67* y se han establecido *5* skipframes para *TFL*.

Aplicando estos parámetros tenemos que, para vídeos con mayor resolución la eficiencia aumenta de manera significativa, frente a los vídeos comprimidos. Esto se debe a que al tener mayor tamaño de imagen, la detección se realiza de mejor manera y con esto dar mejores resultados. En cuanto a la detección de E/S, estudiando la serie de vídeos disponibles, se ha llegado a la conclusión, que mediante el ajuste de los parámetros y aplicando skipframes, se puede llegar a obtener resultados satisfactorios, pero no con ello suficientes dados los objetivos propuestos.

La captura de datos sobre el mapa de calor es realmente satisfactoria, esta, es ajustable dependiendo de la limitación de la memoria disponible, lo que se traduce en coste. Posteriormente se tuvo en cuenta el valor máximo alcanzable por cada celda y el límite superior de píxeles transmitibles a través de un mensaje, que se llegaba capturar para un vídeo de alta resolución. Con esto se ha podido ver cuánta información fue capturada frente a la obtención de información sin filtros y junto con el ajuste de celda para mayor

obtención de detalle frente a rendimiento, tener una perspectiva de los ajustes disponibles del modelo.

Todas estas pruebas quedan limitadas frente a una obtención de información en un entorno real, aunque quede bastante aproximada con una serie de vídeos. Otro de los problemas se debe a grandes grupos de personas y su comportamiento, ya que, únicamente llegaban a juntarse hasta tres individuos al mismo tiempo sobre una zona exacta.

5.3.4. Sensores

A continuación se definen las pruebas realizadas para comprobar el correcto funcionamiento del modelo desarrollado sobre los sensores (ver Tabla 5.13). Todas las pruebas serán realizadas cinco veces para poder disponer de más datos, y con un total de veinte personas a detectar en cada prueba.

ID	Descripción
PIR-01/USO-01	Condiciones normales
PIR-02/USO-02	Una persona entra y otra sale al mismo tiempo
PIR-03/USO-03	Dos personas entran/salen sin espacio entre ellas
PIR-04/USO-04	Una persona permanece quieta frente al sensor y el resto circulan
PIR-05/USO-05	Más de dos personas pasando a la vez
PIR-06/USO-06	Una persona pasa corriendo
PIR-07/USO-07	Interferencias

Tabla 5.13: Pruebas PIR/USO realizadas.

PIR-01/USO-01

Esta prueba se ha realizado bajo condiciones normales, restringiendo el paso a una única persona cada vez.

En este caso, mientras que el modelo con los sensores PIR presenta una tasa de detección del *100 %*, el de sensor ultrasónico tiene un porcentaje de fallos del *30 %*, detectando más gente de la que realmente pasa (ver Figura 5.20).

El motivo de la detección en exceso del sensor ultrasónico, se debe a enviar y recibir una onda de ultrasonidos constantemente. Por ese motivo, si está dependiendo de en qué lugar del cuerpo de la persona rebote, puede llegar a detectarla como más de una persona. El individuo está en movimiento y, si este tiene un brazo por delante del torso, será detectado por el sensor. Al continuar moviéndose la persona, el sensor dejará de detectar al sujeto, sin que este haya llegado realmente a pasar.

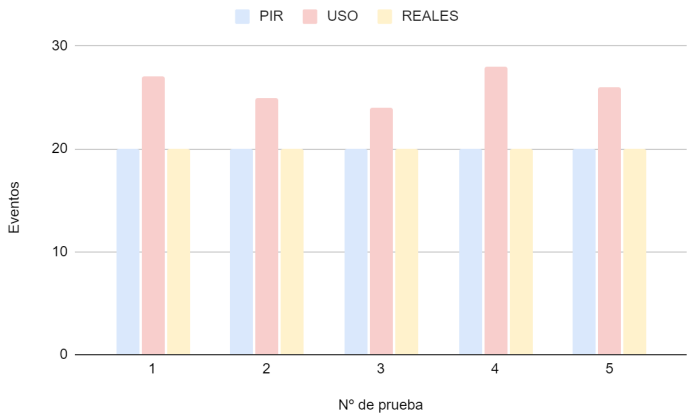


Figura 5.20: Detecciones realizadas bajo condiciones normales.

El torso de la persona será lo siguiente que detecte el sensor, por lo que para este será una nueva detección, por tanto, otra persona. Tras ello, puede darse el caso de que las ondas dejen de rebotar en el torso, pero unos instantes después reboten nuevamente en un brazo.

En el peor de los casos, la situación relatada en los anteriores párrafos se dará, mientras que en el mejor de los casos, las ondas del sensor solamente rebotarán contra la persona en una ocasión. En definitiva, una persona puede ser contabilizada entre una y tres veces.

PIR-02/USO-02

Dado que los sensores van a estar controlando cuánta gente entra y sale por un acceso, se realizó una prueba en la que una persona va a entrar al recinto mientras que otra va a salir al mismo tiempo. Gracias a los datos arrojados (ver Figura 5.21) por las veces que se realizó la prueba, se puede comprobar que ninguno de los dos funciona de la manera esperada, ya que mientras que los PIR detectan con una efectividad del *50 %*, el ultrasónico detecta a un *65 %*.

En el caso de los PIR es entendible, ya que al activarse ambos sensores a la vez y luego desactivarse al mismo tiempo también, solamente detectan uno de los dos casos (es decir, la persona que entra o la persona que sale); y en el caso del ultrasónico pasa algo parecido, ya que al pasar dos personas a la vez por delante del sensor, este solamente detectará a la más cercana. Como se ha explicado en la primera prueba, en algunas ocasiones una persona ha sido contabilizada múltiples ocasiones.

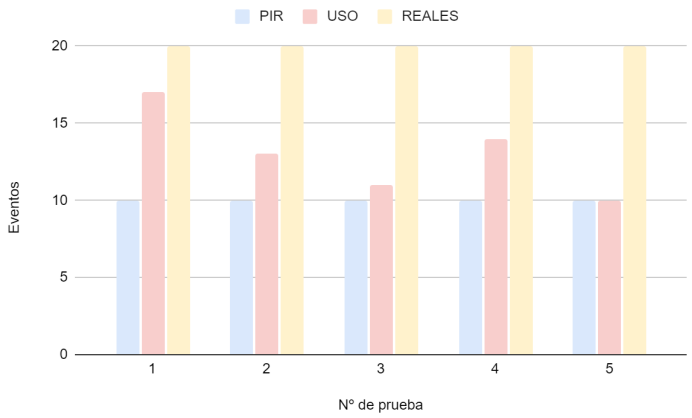


Figura 5.21: Detecciones realizadas cuando un individuo entra y otro sale de manera simultánea.

PIR-03/USO-03

El quid de esta prueba es el hecho de simular que dos personas entren una muy junta de la otra sin dejar apenas espacio entre ellas. En el caso de los PIR, en la Figura 5.22 se puede apreciar que detecta de manera satisfactoria al *50 %*, mientras que, el ultrasónico, vuelve a detectar de más, en este caso un *16 %* extra.

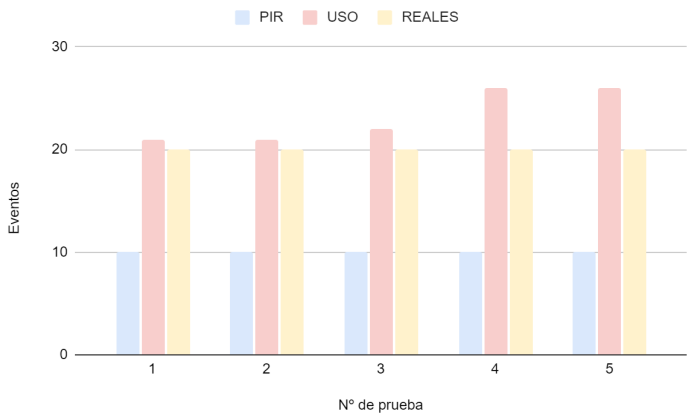


Figura 5.22: Detecciones realizadas cuando dos individuos entran/salen simultáneamente.

El hecho de que el ultrasónico vuelva a detectar de más es por el mismo motivo que lo explicado en la primera prueba (PIR-01 y USO-01).

PIR-04/USO-04

Para esta prueba, una persona ha estado delante de los sensores, y el resto han ido pasando de un lado a otro (un total de *19* veces). Mientras que los PIR están en torno a un *52 %* de acierto, el sensor ultrasónico se sitúa con un extra de un *18 %*, como se puede apreciar en la Figura 5.23.

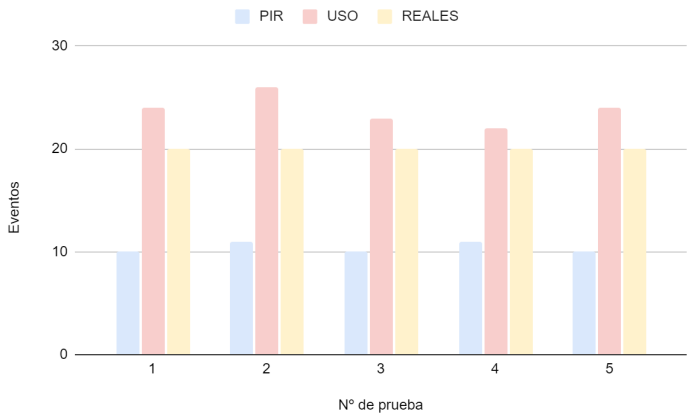


Figura 5.23: Detecciones realizadas cuando un individuo permanece inmóvil, mientras otros circulan.

PIR-05/USO-05

Para esta prueba, entre tres y cuatro personas han pasado por delante de los sensores a la vez, para comprobar si eran capaces de detectar a cada una de las personas o no. En el caso de los PIR, como se activan al detectar radiación infrarroja, independientemente de cuánta gente circule al mismo tiempo solamente se activará una vez; y es por eso por lo que en esta prueba tienen una tasa de acierto del *27 %* (en función de cuánta gente pase a la vez). En cuanto al ultrasónico, el porcentaje de detección es de un *71'2 %* (ver Figura 5.24), aunque debemos tener presente que a una persona puede haberla detectado una o más veces.

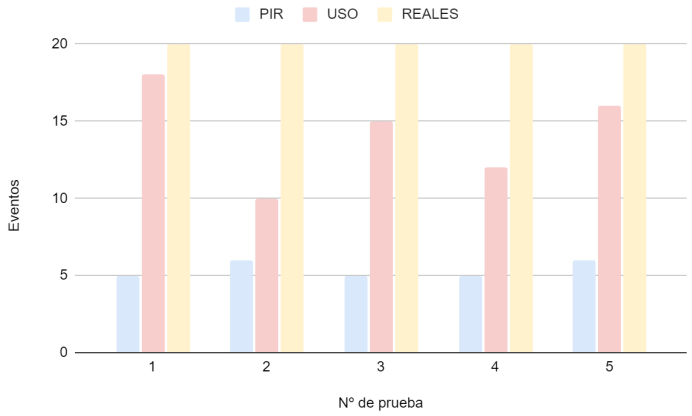


Figura 5.24: Detecciones realizadas cuando más de dos individuos circulan libremente.

PIR-06/USO-06

En esta prueba una única persona ha pasado frente a los sensores corriendo, hasta un total de veinte ocasiones para ver si esta era detectada. Como se puede apreciar

en la Figura 5.25, los PIR tienen un porcentaje de detección del *55 %*, mientras que el ultrasónico es de un *59 %*.

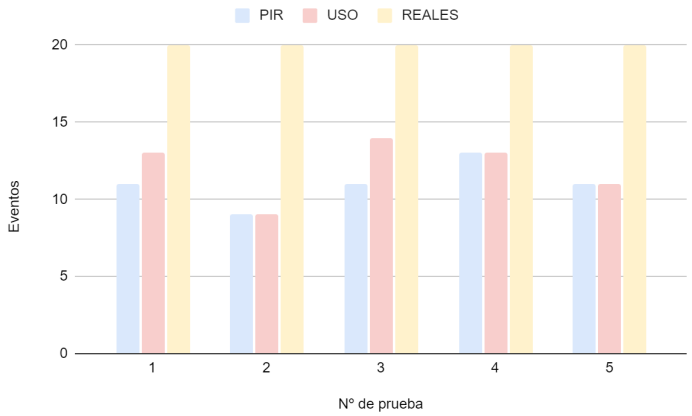


Figura 5.25: Detecciones realizadas cuando hay individuos circulando a un ritmo elevado de velocidad.

PIR-07/USO-07

Se han estado realizando distintas pruebas con distintos objetos emisores de infrarrojos, ya que el sensor PIR es pasivo y capta este tipo de radiación que emiten los cuerpos. Puede dar falsos positivos si hay cerca alguna máquina (por ejemplo, un ordenador) que comience a utilizarse y su temperatura se vea incrementada, con los radiadores pasa lo mismo, y con algunas mascotas. También puede dar falsos positivos si hay interferencias electromagnéticas o radiofrecuencias.

Con respecto al sensor ultrasónico, excluyendo cubetas y máquinas de cavitación (para tratamientos estéticos que consisten en aplicar ultrasonidos en ciertas zonas del cuerpo), la única interferencia captada durante la realización de estas pruebas es que cerca haya otro sensor y se produzcan interferencias entre las ondas, de manera que la lectura sea errónea.

Conclusión

Como se ha podido comprobar, los sensores funcionan bien siempre y cuando el flujo de personas no sea elevado y pasen lo suficientemente separadas como para que sean detectadas individualmente.

La principal desventaja del sensor ultrasónico, es que puede llegar a contar a una misma persona hasta tres veces. De esta manera, aunque en ocasiones esté mostrando una cantidad de personas cercana a la realidad, puede deberse a que alguna de las personas no haya sido detectada, pero dicha detección sea realizada de más mediante la detección múltiple de otra persona.

Su ventaja es que puede llegar a detectar a varias personas en una misma pasada; mientras que el conjunto de sensores PIR no es capaz de hacer dicha distinción. En cambio, este último, funciona perfectamente cuando se trata de detectar personas de una a una.

En un entorno real, al utilizar los sensores estudiados, los datos que estos puedan arrojar han de tomarse como una medición orientativa ya que, como se ha podido visualizar en las pruebas, es altamente improbable que lleguen a dar la cuantía real de personas que hayan circulado por la zona en la que se encuentren instalados.

5.3.5. Modelo

Una vez probados los límites del modelo por separado y ajustados los parámetros para obtener el mayor rendimiento, se van a establecer una serie de pruebas para comprobar la estabilidad de todo el conjunto (ver Tabla 5.14).

ID	Descripción
T-01	Mensajes continuos durante un periodo de tiempo
T-02	Envío de un elevado número de mensajes

Tabla 5.14: Pruebas modelo completo.

T-01

El modelo tiene que estar pensado para poder soportar un uso prolongado. Por esta razón, se ha comprobado un uso moderado del funcionamiento sobre todos los componentes unidos, como se refleja en la estructura descrita en la Figura 4.32.

Una vez realizada la prueba, se enviaron un total de *953* mensajes, de los cuales llegaron un *100 %* de ellos, sin ningún paquete perdido. No se observó ningún error en la red durante la toma de datos, ni en los dispositivos de los extremos.

T-02

La última prueba realizada, consiste en una toma de datos elevados durante un corto periodo de tiempo. Consiste en publicar con todos los dispositivos de manera simultánea una gran cantidad de datos y que, a su vez, el suscriptor obtenga estos datos con el objetivo de comprobar el volumen que puede tramitar.

La toma de datos duró *5* minutos, en el que se pudieron mandar simultáneamente un total de *469* mensajes. Gracias a la cola de mensajes disponible en el broker, podemos procesar ingentes cantidades de datos, aunque por ello se pierda la carga en tiempo real sobre la web.

Como conclusión a este ensayo, tenemos un resultado de un *97,78 %* de mensajes recibidos del total de mensajes enviados. El error producido como pérdida de paquetes, se debe en gran parte al mapa de calor, este genera paquetes fragmentados cada vez que manda la matriz del mensaje. Otros errores reportados son causados por uno de los publicadores de sensores (concretamente el del PIR) y esto se produjo debido a la calidad de servicio asociada a este (QoS 0).

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Una vez terminado el análisis del proyecto se puede concluir que, se ha desarrollado un modelo de bajo coste y consumo, que detecta la ocupación de un espacio cerrado. Los datos se transmiten en el acto, usando el protocolo MQTT, a una aplicación web fácil de comprender para la gran mayoría de usuarios.

El modelo desarrollado cuenta con diferentes programas con los cuales obtener datos, como son la aplicación de reconocimiento y clasificación de imágenes y la aplicación de sensores. Además, hace uso de una aplicación web para exponer de forma visual los datos obtenidos. Todas ellas se conectan mediante Mosquitto, que utiliza el protocolo MQTT. A estas aplicaciones se les han realizado una serie de pruebas para comprobar el rendimiento del modelo, y a la vez, evaluar las diferentes configuraciones que ofrece cada una.

En IoT se ha utilizado QoS *1*, lo que afecta principalmente tanto al retardo de mensajes en la comunicación entre publicador, broker y suscriptor, como a la cantidad de mensajes perdidos cada vez que hay una caída de conexión y esta se recupera.

En cuanto a CV para el código de TFL y el TFL con skipframes, el umbral para la función IOU se estableció a *0,6*, el valor de confianza para la clasificación de objetos a *0,67* (*67 %*), el número total de hilos de procesamiento (fijados en la API) a *16*, y la cantidad de skipframes a *5*. Tanto la cantidad de hilos como la cantidad de skipframes establecidos, influyen en el tiempo de procesamiento de imagen y la tasa de aciertos de la detección.

Respecto a la representación del mapa de calor, se dividió la imagen en celdas de *10* píxeles, formando una matriz a partir del tamaño original. Dependiendo de las dimensiones de estas el tiempo de procesado e impresión del mismo. Se estableció el tamaño citado

para conseguir la menor pérdida de información.

Tras realizar una serie de pruebas con esta configuración se ha logrado obtener el mejor desempeño posible. Esto ha dado lugar a una serie de conclusiones asociadas a los objetivos iniciales.

Relativo al primer objetivo definido en la memoria, se puede decir que se ha desarrollado un sistema por eventos con comunicación inmediata gracias al protocolo MQTT entre las diferentes partes, entendiendo por esto que en el momento en que se produce una detección, el publicador comparte automáticamente el mensaje con el broker y este lo reenvía a la aplicación web. Sin embargo, cabe destacar que la captura de datos realizada por el sistema de vídeo con RPI, no llega a tomar muestras eficientes de vídeo en tiempo real, según se ha podido comprobar con las diferentes pruebas y es por esto, que no ha quedado cumplido el objetivo de manera satisfactoria.

Otro de los puntos a tener en cuenta es el mayor rendimiento dentro de un bajo coste. Este, se ha aproximado lo máximo posible a lo esperado tanto para detección de vídeo con TFL, como para tener el resto del sistema. Cabe destacar, que al ser una aplicación web, podría montarse el servidor web sobre otra RPI, ahorrando el coste del alojamiento en servidor externo y como se pudo ver en Sección 2.4, el coste total se encuentra alrededor de un *10 %* menos del valor de mercado.

Además, la facilidad de uso para la lectura e interpretación se ha cumplido con creces. La web ofrece diversas posibilidades que cualquier usuario, sin conocimiento previo, puede entender de manera sencilla. Por otra parte, gracias a ser una aplicación compilada con diseño adaptativo, esta se puede reproducir en diferentes dispositivos sin necesidad de instalar ninguna aplicación adicional a la del navegador.

En cuanto al control de ocupación mediante mapas de calor, se ha llegado a enviar y reproducir de cara a un usuario final, de manera satisfactoria. Gracias a esto, se podrían hacer estudios de comportamiento, teniendo un rol de administrador, sin necesidad de obtener los datos directamente del publicador. Por último, tras la serie de pruebas realizadas, se ha podido procesar todo el conjunto del modelo con una gran estabilidad gracias a las librerías de MQTT empleadas y debidamente configuradas.

Una vez visto el cumplimiento de objetivos, se debe mencionar cómo los sensores obtienen datos sin apenas retardo, pero con grandes fallos a la hora de capturar el tráfico generado en numerosos grupos de personas. Mientras que la captura con CV no llega a obtener el control de aforo en tiempo real. Previamente se ha explicado que si se utiliza el código con TFL, la pérdida de información no es muy notable incluso teniendo E/S bidireccional o unidireccional, pero para poder tener un sistema en tiempo real se precisa de

TFL con skipframes, en el que se pierde gran cantidad de la información requerida.

El fallo de este modelo, al ser de tan bajo coste, se ha producido al no invertir en aceleradores de tipo TPU[1], los cuales hubieran permitido emplear RRNN con pesos *float32* y así tener un sistema en tiempo real más preciso, con un coste adicional necesario. Otra opción que podría solventarlo, sería estudiar si Nvidia Jetson Nano podría sustituir al conjunto de RPI con una TPU. Otro de los problemas asignados a las pruebas realizadas, es la falta de material para tomar datos, esto es, el modelo no se pudo tomar en un entorno real ni se pudieron conseguir vídeos con los que sacar mejores conclusiones.

6.2. Trabajo Futuro

Esta sección trata sobre posibles trabajos futuros que podrían realizarse en adición al trabajo desarrollado en este proyecto:

- **Accesos unidireccionales:** actualmente el modelo soporta única y exclusivamente accesos que sean bidireccionales, es decir, que al recinto controlado por el modelo desarrollado se puede entrar y salir por los mismos accesos. Se planteó de esta manera dado que la mayoría de recintos tiene accesos por los que entrar y salir, de manera que internamente el modelo gestiona cada dispositivo (en este caso, refiriéndose a cámaras o sensores) conectado a él de forma que contabilice el flujo de personas que circulen en ambos sentidos. Para este propósito se debería realizar un ajuste en el modelo general, de manera que así ya se podrían conectar nuevos sensores o cámaras y solamente manden datos de entrada o de salida. De igual forma, en el caso de los sensores, no sería necesario realizar este cambio. Bastaría con emplear dos NodeMCU, de manera que una hace de cliente y controla un acceso, y la otra además de controlar otro acceso actúa como servidor, por lo que esta última sería la que se comunicaría con el broker y le mandaría sus datos junto con los de la NodeMCU cliente.

- **Sincronización de todas las aplicaciones:** pese a que todas las aplicaciones funcionan correctamente de manera simultánea, en un futuro se podría mejorar la sincronización de los datos de todas ellas para hacer que los resultados que estas proporcionan sean aún más precisos. Para ello bastaría con hacer que todas las aplicaciones recibieran los datos del resto y comprobaran si tienen los mismos, en caso de no ser iguales se escogerían los datos de la aplicación web, ya que son los datos finales.

- **Aceptar múltiples zonas y mapas de calor:** como implementación futura se ha

[1]Unidad de procesamiento tensorial

planteado que la aplicación web sea capaz de mostrar más de una zona para poder instalar varias cámaras y, de esta forma, controlar un espacio de forma más precisa. Esto mismo podría aplicarse a los mapas de calor haciendo que se muestre más de un mapa.

- **Implementación de varios roles de usuario:** en una futura versión de la aplicación web podrían incorporarse varios tipos de usuarios, por lo que se necesitaría implementar diferentes tipos de roles que otorguen acceso a ciertas partes de la aplicación.
- **Cajas delimitadoras 3D:** uno de los problemas que acontecen en el control de aforo usando RRNN, se debe al uso de funciones como NMS. La razón del uso de estas funciones se encuentra en la unión de diversas BB, causando equivocación con dos personas situadas en línea a la visión de la cámara.

Figura 6.1: 3D Bounding Box [3].

Como se puede observar en la Figura 6.1, la manera de establecer los límites de un objeto, define una profundidad adecuada a esta. De este modo e implementando una función NMS, las BB podrían aproximarse en mayor medida.

- **Uso de aceleradores:** basando el estudio en las conclusiones expuestas, hay una gran carencia en cuanto al objetivo de detección en tiempo real. Por lo que, se propone repetir el mismo estudio, aplicando aceleradores gráficos como pueden ser las TPUs. Tanto para el propio modelo propuesto, como uno que emplee BB en tres dimensiones, es necesario un alto incremento en el rendimiento.

Bibliografía

[1] P. F. S.A., "Sensor infrarrojo de movimiento pir hc-sr501." URL https://www.puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf, 2017.

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[3] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang, "Gs3d: An efficient 3d object detection framework for autonomous driving," pp. 1019–1028, 06 2019.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[5] V. Martínez Fuentes, "Introducción a la plataforma arduino y al sensor ultrasónico hc-sr04: experimentado en una aplicación para medición de distancias," B.S. thesis, Universidad Carlos III de Madrid, 2016.

[6] C. D. Ortega and F. E. Moyano, "Técnicas de implementación de visión estereoscópica en robótica," in *XVI Concurso de Trabajos Estudiantiles (EST)-JAIIO 42 (2013)*, 2013.

[7] DescubreArduino.com, "Esp32 vs esp8266 ¿cuáles son las diferencias entre ambos módulos?." URL https://descubrearduino.com/esp32-vs-esp8266//, 2021.

[8] J. M. Ponce Real *et al.*, *Computer Vision in Oliviculture. Contributions to the Post-harvest Estimation of Individual Fruit Features, Early In-the-field Yield Prediction, and Individual Tree Characterisation from Aerial Imagery, by means of Image Analysis.* PhD thesis, Universidad de Huelva, 2020.

[9] E. I. Safonov and P. V. Zavialov, "Development of software and hardware of a robot using computer vision algorithms," *Yugra State University Bulletin*, vol. 16, no. 4, pp. 33–41, 2020.

[10] R. A. AlQadi, A. Zaghloul, and S. A. Taie, "An occupancy-based strategy employing computer vision for reducing cooling energy consumed in buildings," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 3, 2021.

[11] The Apache Software Foundation, URL https://wicket.apache.org/learn/, *Apache Wicket - All the things you want to know about Wicket but are afraid to ask*, 2021.

[12] A. Vukotic and J. Goodwill, *Apache Tomcat 7*. Springer, 2011.

[13] L. A. C. Santillán, M. G. Ginestà, and Ó. P. Mora, "Bases de datos en MySQL," *Universitat oberta de Catalunya*, 2014.

[14] J. E. Duque Parra, J. Barco Ríos, and F. J. C. Peláez Cortes, "Santiago felipe ramón y cajal, ¿padre de la neurociencia o pionero de la ciencia neural?," *International Journal of Morphology*, vol. 29, no. 4, pp. 1202–1206, 2011.

[15] D. J. Matich, "Redes neuronales: Conceptos básicos y aplicaciones," *Universidad Tecnológica Nacional, México*, vol. 41, pp. 12–16, 2001.

[16] A. Simón Rodríguez and R. Ruz Gómez, *Evaluación de algoritmos de machine learning para conducción*. PhD thesis, Universidad Complutense de Madrid/Universidad de Granada, 2020.

[17] Á. Moreno Prieto, *Detección de objetos con TinyYOLOv3 sobre Raspberry Pi 3*. PhD thesis, Universidad de Sevilla, 2020.

[18] J. M. S. Arturo Barbero Pérez, Alejandro Cabezas Garríguez, *Facial Detection On Digital Videos*. PhD thesis, Facultad de Informática, Universidad Complutense de Madrid, 2020.

[19] A. Rosebrock, "Simple object tracking with opencv," *PyImageSearch*, jul 2018.

[20] N. Segura Gordillo, *Marketing del color ¿Cómo influye el color del logotipo en la personalidad de una marca?* PhD thesis, Universidad de Chile, 2016.

[21] M. Sandra Cuervo Diez, *El poder del color. La influencia de los colores en los consumidores.* PhD thesis, Facultad de Ciencias Económicas y Empresariales - Universidad de León, 2012.

[22] G. saimj7, "saimj7/people-counting-in-real-time." License: MIT.

[23] Raspberry Pi Org, URL https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/, *Raspberry Pi 4 Model B specifications*, 2021.

[24] C. 101, *NodeMCU ESP8266 Pinout, Specifications, Features & Datasheet*, 2020.

[25] L. Carmenate Milián, F. A. Moncada Chévez, and E. W. Borjas Leiva, *Manual de medidas antropométricas*, 2014.

[26] J. Dagnino, "Coeficiente de correlacion lineal de pearson," *Chil Anest*, vol. 43, pp. 150–153, 2014.

[27] M. Rapoport, *La globalización económica*. PhD thesis, Facultad de Ciencias Económicas. Universidad de Buenos Aires, 2013.

[28] The ZeroMQ community, URL https://zeromq.org, *ZeroMQ*, 2007.

[29] Apache Software Foundation, URL https://activemq.apache.org, *ActiveMQ*, 2004.

[30] V. Arévalo, J. González, and G. Ambrosio, "La librería de visión artificial opencv. aplicación a la docencia e investigación," *Base Informática*, vol. 40, pp. 61–66, 2004.

[31] I. García and V. Caranqui, "La visión artificial y los campos de aplicación," *Tierra infinita*, vol. 1, no. 1, pp. 94–103, 2015.

[32] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[33] C. Bauer and G. King, *Hibernate in action*, vol. 1. Manning Greenwich CT, 2005.

[34] M. Trigás Gallego, *Metodologia scrum*. PhD thesis, Universitat Oberta de Catalunya, 2012.

[35] Hibernate org., URL https://hibernate.org/orm/, *Your relational data. Objectively. - Hibernate ORM*, 2021.

[36] P. Siriwardena, *Maven Essentials*. Packt Publishing Ltd, 2015.

[37] K. Gurumurthy, *Pro wicket*. Apress, 2007.

[38] J. J. Gutiérrez, "¿qué es un framework web?," *Available in: http://www. lsi. us. es/˜ javierj/investigacion_ficheros/Framework. pdf Accessed May*, vol. 12, 2014.

[39] J. Gosling, B. Joy, G. Steele, and G. Bracha, *The Java language specification*. Addison-Wesley Professional, 2000.

[40] R. Priyadarshini and R. Mehra, "Quantitative review of occupancy detection technologies," *Int. J. Radio Freq*, vol. 1, pp. 1–19, 2015.

[41] Standard, OASIS, URL http://docs.oasis-open.org/mqtt/mqtt/v3, *MQTT version 3.1.1*, 2014.

[42] F. Moreno Cerdà, "Demostrador arquitectura publish/subscribe con mqtt," B.S. thesis, Universitat Politècnica de Catalunya, 2018.

[43] Standard, OASIS, URL https://mqtt.org, *The Standard for IoT Messaging*, 1999.

[44] Eclipse Fundation, URL https://mosquitto.org, *Eclipse Mosquitto*, Jan 2018.

[45] T. Castro Blanco, "Sistema de medida de temperatura basado en nodemcu y android," B.S. thesis, Universidad Carlos III de Madrid. Departamento de Ingeniería Eléctrica, 2018.

[46] L. Llamas, "Nodemcu, la popular placa de desarrollo con esp8266." URL https://www.luisllamas.es/esp8266-nodemcu/, 2018.

Acrónimos

AJAX Asynchronous JavaScript and XML. 20, 35, 107

BB Bounding Box. 31, 33, 47, 48, 56, 57, 71, 72, 89, 106

CNN Convolutional Neural Networks. 28

CT Centroid Tracker. 33, 47, 49

CV Visión por Computador. 14, 29, 78, 86, 87, 103, 104, 110

E/S Entrada-Salida. 3–5, 18, 47, 57, 69, 75, 76, 78, 87, 98, 99, 110

FRCNN Faster R-CNN. 28, 56, 57

IA Inteligencia Artificial. 11, 14, 17, 26, 57

IoT Internet of Things. 4–6, 22, 23, 86, 98, 99, 101, 103, 112

IOU Intersection Over Union. 31, 32, 46, 71, 72, 78, 86, 103

JAR Java Archive. 21

JDK Java Development Kit. 108

JSON JavaScript Object Notation. 39

MAC ID Media Access Control Identifier. 8

MITL MIT License. 56

MQTT Message Queuing Telemetry Transport. 15, 16, 22, 23, 25, 44, 45, 53, 57, 86, 87, 101, 103, 104, 109

NMS Non-Maximum Suppression. 31, 33, 47, 57, 89, 106

OpenCV Open Source Computer Vision. 14, 30, 31, 56, 57, 109

ORM Object/Relational Mapping. 20

PIR Sensor de Infrarrojos Pasivo. 3, 9, 10, 17, 51–55, 57, 79–85, 98, 110, 111

PY Python. 44, 56

QoS Quality of Service. 24, 63, 64, 66–68, 86, 103

ReLU Función Lineal Rectificadora. 46

RPI Raspberry Pi. 3, 6, 29, 30, 47, 57, 69, 72, 77, 87, 88, 98, 101, 104, 105, 109, 111

RPN Regional Proposal Networks. 28, 29

RRNN Redes Neuronales. 14, 26, 29–31, 69, 70, 73, 88, 89, 104, 106

SIGMOID Función Sigmoide. 46

SQL Structured Query Language. 21

SSD Single Shot Detector. 28–30, 46, 69, 71, 73

TF TensorFlow. 15, 57

TFG trabajo de fin de grado. 4, 99

TFL TensorFlow-Lite API. 6, 14, 46, 57, 69, 72, 73, 86–88, 101, 103, 104, 109, 110

URL Uniform Resource Locator. 35, 39

WAR Web Application Archive. 21, 108

XML Extensible Markup Language. 21

YOLO You Only Look Once. 28, 29

Apéndice A

Introduction

A.1. Motivation

Throughout the last century, human population has increased exponentually, and it is expected to continue growing even more. This contributes to the existence of multitudinous events, large influx in certain places and the growth of urban areas thanks to digital transformation.

In the context in which this project has been developed, COVID-19 pandemic has caused several problems in today's society in many aspects. This has led to the search and implementation of some solutions to try to avoid them, attempting to minimize the impact on day-to-day life as much as possible. However, it has been inevitable to have to make changes in aspects as basic as mobility for sake of safety.

This contributes to the need to have an occupancy control. Although this is not a new idea [1], it must be admitted that it was not as widespread as it is nowadays, and has had a great boom in recent years. Despite this, most used models are usually expensive, or consists of one person at the entrances to the premises, counting the people who enter and leave.

For all of this, it was proposed to create a real-time control system of people in closed spaces, being affordable and automatic. These two main concepts were the ones that drove the realization of this work. Thanks to technological advances, this system can be improved, made cheaper, and made available to general public.

[1] https://www.elindependiente.com/tendencias/2019/12/30/puerta-sol-blinda-campanadas-cuatro-filtros-policiales-maximo-19000-personas/

A.2. Object of the Investigation

A series of guidelines were followed to develop this project, and thus, being able to draw a conclusion based on them. To fulfill the task, the main objective can be subdivided into the following ones:

- The system must work in real time, which implies that the environment is continuously being monitored, and when a new event is detected, must be captured and transmitted at the precise moment in which occurs.
- The system must have the highest performance within what is feasible, sticking to a low cost. It is intended to obtain the highest possible profitability and that this adjusts to the rest of the objectives
- To facilitate its use, the system must have a graphical interface and be multi-device. This implies that the data collected must be accessible and easy to view. And so, for example, the data of a premises can be displayed for its administrators.
- Occupation within a closed area must be known at all times. Not only that, but also heat maps of the different areas of the same can be made, so that the places where there is a greater capacity are known.
- The stability of the system must be guaranteed, so that the data does not cease to be accessible, or a component may be affected in the event that there is a huge amount of data to be collected and sent.

A.3. Workplan

This project began to be developed in July 2020. As a model, an agile work methodology was followed, since it seemed convenient to have meetings between the students every little time. In this way, the opportunity was given to be able to establish a common roadmap for the established deadlines and sharing the progress and concerns that were had. Similarly, each month there was a meeting between students and tutors, to check the work done and thus be able to verify that these advances were on the right track or if they should make any changes.

To understand the chronology, the work developed by the students can be grouped into the following stages:

First stage - Previous work

This stage is made up of the time elapsed from the first meeting between the students and the tutors, until the middle of August. During those two months,

an investigation was carried out prior to the beginning of the project, in which different systems for detecting people were searched. In this way, the project could be channeled towards the most convenient route, in this case, the one that met the objectives described in the previous point.

Second stage - Development

This stage focuses on the implementation of the different parts of the model, in terms of the paths taken simultaneously and the actions carried out.

The development of the web application was focused on the creation of different functionalities, in such a way that several roles were allowed and all the design elements had a *Responsive Web Design* character. Thus, the web could be adapted to different devices and it would be possible to interact with it anywhere.

The detection of E/S that was implemented on the detection part was focused on improving the code, so that it could be executed in an RPI or a device with similar characteristics. The main idea of this part was to improve the usefulness of this detection, without losing efficiency in the results.

Following the development, it is necessary to talk about sensors and communication devices, which had to be understood first before being able to acquire them. It began by doing tests, in a sub-stage, with the sensor PIR (half stage), which would not be continued until the model was almost completely finished.

Synchronously to the previous sub-stages, the necessary modules to establish communication were developed. Functions were established within each program, to be able to communicate the different languages that were used in the rest of the modules. Prior to this, it was necessary to conduct a study on the available options, as well as different tests until the communication method was properly adjusted.

Third stage - Documentation

The documentation stage was carried out at various points during the project. To begin to carry out the work, the students had to prepare adequately, for this, they proceeded to document the findings and the conclusions that were reached.

Occasionally, some notes were made to remember how certain software installations or momentous advances had been made for the project. However, it was not until the end of the development stage that memory was emphasized again. In this way, further progress could be made in the project, whether in the field of web development, heat map, E/S detection with video, IoT or E/S detection with sensors.

When only minor corrections to the project remained, work continued on the do-

cumentation. Meanwhile, a series of tests were carried out on the model developed by the students to determine the correct functioning of what was developed.

Fourth stage - Tests

As the development part progressed, tests were carried out to verify the operation of the proposed model and check with which settings it would be more optimal. Gradually the tests were intensified to be able to obtain a massification of the resulting data to be compared with each other, in order to prioritize some advances over others. At the end of the full development, resistance tests were carried out to check the stability of the system.

It is important to define it more visually, this is why a Gantt chart (see Table A.1 and Figure A.1) is added, showing the previous phases and the most notable milestones.

A.4. Structure of the Work

To simplify the understanding of the work carried out by the group, this report has been written sequentially. It is divided into six chapters and four appendices.

In the first chapter, being this one the last section, the main motivations and objectives that have identified this final degree project are explained. In addition to the work plan followed, ensuring the optimal management of available resources.

In the second one, a study of the technologies that are used today will be carried out. With this we will compare the different hardware on which the TFG could be based, bearing in mind the objectives set. Likewise, an example will be presented on the subject that is being discussed.

Subsequently, the technologies that have been used to build the model will continue to be listed and explained, and so an explanation of them will be given in the third chapter. Specifically, it will be subdivided according to the category in which they have been used. In this way, there will be left with three main groups: web, IoT and artificial intelligence (used to detect the E/S of an enclosed space and the corresponding calculations to generate a heat map).

The fourth chapter will describe the structure of the model, the scenarios in which it can be applied and its scalability. Performance and other relevant aspects of it will also be analyzed.

Next, in chapter five, the limits of the project will be defined and a battery of tests will be carried out to help document and justify them.

Section six will contain the conclusions reached after the project and possible improvements or ways in which it would be possible to advance and expand the work, in the future, with new features or increase its functionality, will be pointed out.

The seventh and eighth chapters will be the first and the sixth translated into English, so that the conclusions that have been reached by carrying out this work reach a wider audience.

Finally, there can be finded appendices where information that seems relevant to highlight will be expanded, such as the work done by each member.

Identifier	Milestone
A	*Completion of previous work*
B	*Development*
B1	Completion of Python development
B2	C++
B21	First functional code
B22	First full test on RPI
B23	TFL compilation on RPI
B24	Heat map completion
B25	Implemented skip frames
B3	IoT
B31	Search of information about MQTT
B32	Comparison of different methods
B33	Search of data in different programming languages
B34	Broker monitoring and settings
B35	Send of messages improving scalability
B36	Send a data matrix to represent the heat map
B37	Send of an image on which the heat map will be painted
B40	Sensors
B41	Model developed for PIR
B42	Connection of sensors with NodeMCU, and this one to the broker
B43	Model made for ultrasonic sensor
B5	Web
B51	Home page created
B52	Communication with broker
B53	Real-time capacity updating
B54	Created DB Tables, communication and persistence
B55	Created page with graphics
B56	Security for users added
B57	Receive and display heat map
B58	Define of configurable parameters and how to edit them
B59	Layout and design
B60	Tests and bug resolution
C	*Documentation*
C1	Initial research documentation prior to the start of the project
C2	Documentation of various advances
C3	Project memory made
D	*Tests*
D1	Python
D2	C++
D3	IoT
D4	Sensors
D5	Web
D6	Combined

Table A.1: Milestones of the project during the process.

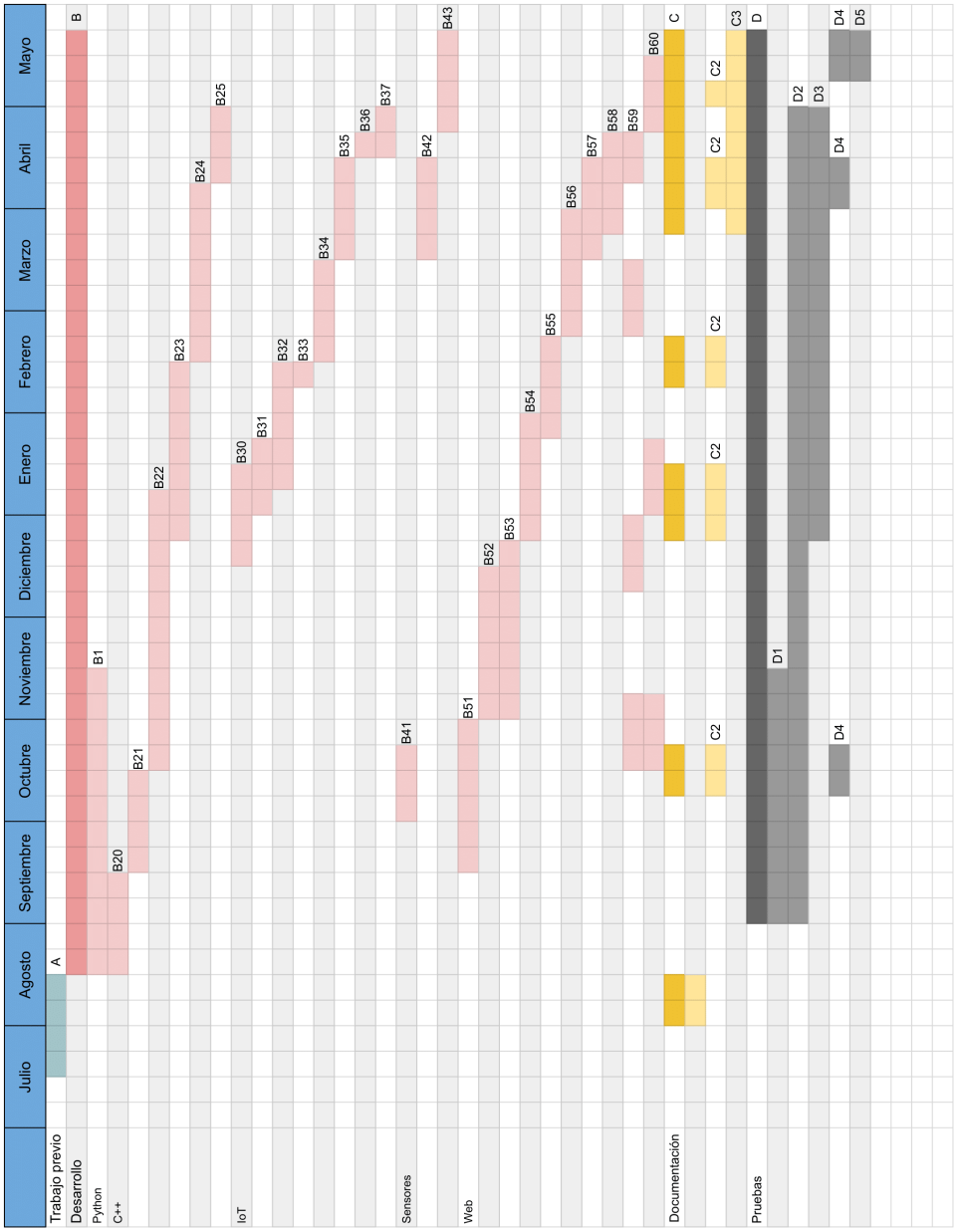


Figure A.1: Gantt diagram followed by the team.

Apéndice B

Conclusions and Future Work

B.1. Conclusions

Once the analysis of the project has been completed, it can be concluded that a low cost and low consumption model has been developed, which detects in real time the occupation of an enclosed space and transmits the data simultaneously, using the MQTT protocol for this, to a web application easy to understand for the vast majority of users.

The model has different programs with which data is obtained, such as the image recognition and classification application and the sensor application. In addition, it makes use of a web application to visually display the data obtained. All of them are connected through Mosquitto, which uses the MQTT protocol. A series of tests have been carried out on these applications to check the performance of the model and, at the same time, to evaluate the different configurations offered by each one.

For IoT it has been used QoS *1*, which mainly affects both the message delay in the communication between publisher, broker and subscriber, and the number of messages lost every time there is a connection drop and the connection recovers.

Regarding CV, for the TFL code and the TFL with skipframes, the threshold for the IOU function was set to *0.6*, the confidence value for object classification to *0.67* (*67*), the total number of processing threads (set in the API) to *16* and the number of skipframes to *5*, which influence the image processing time and detection hit rate.

Regarding the representation of the heat map, the image was divided into cells of *10* pixels, forming a matrix from the original size. The processing and printing time depended on the dimensions of these cells. The size was set to achieve the least loss of information.

Concerning the representation of the heat map, the image was divided into cells of *10* pixels, forming a matrix from the original size. The processing and printing time depended on the dimensions of these cells. The size was set to achieve the least loss of information.

After performing a series of tests with this configuration, the best possible performance has been obtained. This has led to a series of conclusions associated with the initial objectives.

Regarding the first objective defined in the memory, it can be said that an event system has been developed with immediate communication thanks to the MQTT protocol between the different parts, understanding by this that when a detection occurs, the publisher shares automatically the message with the broker, and the broker forwards it to the web application. On top of that, it should be noted that the data capture carried out by the video system with RPI does not manage to take efficient video samples in real time, as has been verified with the different tests.

In addition, the ease of use for reading and interpretation has been more than fulfilled. The website offers several possibilities that any user, without previous knowledge, can understand in a simple way. On the other hand, thanks to being an application compiled with adaptive design, it can be reproduced on different devices without the need to install any additional application other than the browser.

As for the occupancy control by means of heat maps, it has been sent and reproduced, facing an end user, in a satisfactory way. Thanks to this, behavioral studies could be made, having an administrator role, without the need to obtain the data directly from the publisher. Finally, after the series of tests carried out, it has been possible to process the whole model with great stability thanks to the MQTT libraries used and properly configured.

Having seen the fulfillment of objectives, it should be mentioned how the sensors obtain data with hardly any delay, but with large failures when it comes to capturing the traffic generated in numerous groups of people. While the capture with CV does not manage to obtain the capacity control in real time. Previously it has been explained that if the code is used with TFL, the loss of information is not very noticeable even with bidirectional or unidirectional TFL, but in order to have a real-time system it is necessary to use TFL with skipframes, in which a great amount of the required information is lost.

The failure of this model, being so low cost, has been produced by not investing in TPU accelerators, which would have allowed to use RRNN with *float32* weights and thus have a more accurate real time system, with an additional necessary cost. Another option that

could solve it, would be to study if Nvidia Jetson Nano could replace the RPI array with a TPU. Another of the problems assigned to the tests carried out is the lack of material to take data, that is, the model could not be taken in a real environment nor could videos be obtained with which to draw better conclusions.

B.2. Future Work

This section deals with possible future work that could be done in addition to the work developed in this project:

- **One-way access:** currently the model supports only and exclusively accesses that are bidirectional, that is to say, the enclosure controlled by the developed model can be entered and exited through the same entrances. It was proposed in this way given that the majority of enclosures have accesses through which to enter and exit, so that internally the model manages each device (in this case, referring to cameras or sensors) connected to it in such a way that it counts the flow of people through both ways. For this purpose, an adjustment should be made in the general model, so that new sensors or cameras could already be connected and only send input or output data. Similarly, in the case of sensors, it would not be necessary to make this change. It would be enough to use two NodeMCUs, so that one acts as a client and controls an access, and the other, in addition to controlling another access, acts as a server, so the latter would be the one that would communicate with the broker and send it its data along with those of the NodeMCU client.

- **Synchronization of all applications:** although all the applications work correctly simultaneously, in the future the data synchronization of all of them could be improved to make the results they provide even more accurate. For this, it would be enough to make all the applications receive the data of the rest and check if they have the same, if they are not the same, the data of the web application would be chosen, since they are the final data.

- **Accept multiple zones and heat maps:** as a future implementation, it has been proposed that the web application should be capable of showing more than one area in order to install several cameras and, in this way, control a space more precisely. The same could apply to heat maps by displaying more than one map.

- **Implementation of multiple user roles:** in a future version of the web application, several types of users could be incorporated, so it would be necessary to implement different types of roles that grant access to certain parts of the application.

- **3D bounding boxes:** one of the problems that occurs in the capacity control using RRNN, is due to the use of functions like NMS. The reason for the use of these functions is found in the union of different BB, causing a mistake with two people located in line with the vision of the camera.

  As can be seen in Figure B.1, the way to establish the limits of an object defines an appropriate depth for this. In this way and implementing a NMS function, the BB could be approximated to a greater extent.

Figure B.1: 3D Bounding Box [3].

- **Use of accelerators:** basing the study on the above conclusions, there is a great lack in terms of the real-time detection objective. Therefore, it is proposed to repeat the same study, applying graphic accelerators such as TPUs. Both for the proposed model itself, and one that uses BB in three dimensions, a high increase in performance is necessary.

Apéndice C

Reparto de trabajo

Miguel Artell Moreno

Mi aportación al modelo desarrollado se ha centrado principalmente en el módulo de la aplicación web.

Primero me enfoqué en la búsqueda y comparación de diferentes interfaces que ya estuviesen siendo usadas. De esta forma, cogí varias ideas y las expuse a mis compañeros para crear, entre todos, bocetos de la aplicación con las funcionalidades que debían ser implementadas.

Con el crecimiento de tareas a realizar, para ayudar con la organización y gestión, creé un repositorio de GitHub y di de alta un espacio para nuestro equipo en la aplicación Jira Software[1]. Expliqué a mis compañeros el funcionamiento de la misma y comenzamos a usar la metodología SCRUM [34] que hemos seguido para el desarrollo del modelo.

Tras sopesar distintas tecnologías web y decidirme por las ya explicadas, creé una aplicación sencilla usando Maven, la cual ha ido creciendo hasta convertirse en la actual. Para esto seguí una serie de etapas.

Primero programé la página principal. Busqué el tipo de gráfico para indicar la ocupación más adecuado y, siguiendo los bocetos, le di forma. Luego, junto a Víctor, que ha desarrollado la parte de la conexión al broker, realizamos un procesamiento básico de los datos. De esta forma, añadí comportamientos AJAX (utilizando el framework Wicket) para que se fuesen visualizando las lecturas tomadas por la parte de conteo en la web.

Una vez la página principal fue funcional, invertí algún tiempo en modificar la apa-

[1] https://www.atlassian.com/es/software/jira (del grupo Atlassian)

riencia, pasando por muchos diseños diferentes y viendo cual se adecuaba más a lo que estábamos buscando. En este contexto, Jesús y yo diseñamos el logo de la aplicación para presentarlo al resto del grupo y que nos diesen su visto bueno. Posteriormente, se definieron las diferentes pestañas que tendría la web e implementé la barra de navegación.

La siguiente página en ser creada fue la de configuración, debido a la necesidad que nos surgió de controlar la conexión con el bróker desde la web. Eventualmente, amplié esta parte con el resto de las opciones editables explicadas en apartados anteriores. Al plantear y crear el archivo JSON, para poder guardarlas y utilizarlas sin necesidad de conexión con la base de datos, surgió el problema de leerlo desde el servidor Tomcat. A encontrar la ruta adecuada para hacerlo me ayudó Víctor.

Al mismo tiempo, se planteó realizar una persistencia de los datos que llegaban del broker para mostrarlos en gráficas. Con esta idea, Daniel y yo definimos las tablas de la base de datos y las creamos.

Para la persistencia y recuperación de datos, tuve que leer diferente documentación y contemplar distintas opciones hasta que decidí usar Hibernate. Debido a esto, creé las clases necesarias e hice diferentes pruebas para comprobar que todo funcionaba correctamente.

Una vez acabado el trabajo de persistencia, añadí a la web la página de estadísticas. En este caso, Daniel buscó las opciones de visualización que debían tener los gráficos en Java, para que pudiesen mostrarse, y posteriormente, yo creé la gráfica semanal y él quedó a cargo de la gráfica diaria. Sin embargo, en las últimas pruebas globales del modelo, nos dimos cuenta de que las fechas no se estaban generando correctamente y de que el gráfico diario no tenía el comportamiento esperado, cosa que corregí.

Más adelante, cuando Jesús pudo generar datos para el mapa de calor, implementé la página destinada a visualizarlo y le añadí un comportamiento automático para que se fuese refrescando y aplicando el gradiente (definido por Jesús). Al mismo tiempo, metí la seguridad en la web, para que cada vez que alguien intentase acceder a una parte protegida sin estar identificado, fuese redirigido a la página de inicio de sesión.

En este punto, surgieron errores en el despliegue del WAR en Linux. Hice un poco de investigación y cambiando algunas dependencias y las versiones de Java Development Kit (JDK) y Tomcat conseguí solucionar el problema, consiguiendo así que funcionase el servidor web en Linux.

Por último, añadí una pantalla con información del proyecto y del equipo de trabajo y realicé las pruebas que han sido expuestas en el apartado destinado a ello, además

de ayudar con la redacción y documentación de la memoria.

Jesús Álvarez Coll

Para comenzar a explicar cuál fue mi aportación, he de recalcar no sólo la partición individual, sino también la colectiva. En primer lugar, se realizó una investigación sobre qué tipos de sistemas de control de aforo estaban disponibles en la actualidad, del que fui partícipe. Más tarde, se comenzaría una búsqueda de código inicial del cual partir, en la que pude ver cómo funcionaba un código modelo (código escrito en Python), para posteriormente aplicar dicho código en un lenguaje estático siguiendo el programa original.

Una vez hice funcionar el código de partida, mi compañero Víctor me ayudó a ver las diferentes configuraciones para sacarle el mayor rendimiento obtenido. Esta tarea dedicó mucho tiempo, ya que, estaría probando códigos y variaciones de la herramienta hasta conseguir mejores resultados. Ulteriormente, procedí a comparar dicha investigación con la que realizó mi copartícipe Víctor, y así, ver con qué programa se podría continuar.

Compilé la librería de OpenCV en la RPI con la finalidad de probar el código sobre este dispositivo con procesador *ARM*[2], el cual fue probado con anterioridad sobre un procesador *x86*.

Durante la investigación, se probaron lenguajes compatibles con la herramienta, se estudiaron las librerías usadas con la intención de obtener el mayor rendimiento. Una vez cesadas las pruebas sobre el código escrito en C++, procedí a aplicarle un perfilado para comprobar qué función sobrecargaba más la aplicación y, de este modo, poder sustituirla o mejorarla.

Teniendo en cuenta que los resultados obtenidos, dieron lugar al análisis sobre los resultados relacionados a la detección y clasificación de objetos. A partir de ahí, opté por otras opciones estudiadas con anterioridad. A continuación, comencé a investigar la eficiencia que podría alcanzar la librería de TFL.

Una vez hice funcionar el código, tuve que probar la eficiencia de este sobre la RPI y de esta manera, pude observar la gran diferencia entre ambos. De igual forma, hubo que compilar las librerías como en el anterior código aunque, por el contrario, esta tarea pudo llevarme más tiempo del requerido.

Otra de las tareas que comencé, fue el estudio de la comunicación de los sensores a través de MQTT. Esto generó una serie de problemas asociados a la alimentación del pir que tuve que suplir no sólo estudiándo los NodeMCU, sino también comprando

[2]Advanced RISC Machine.

módulos que no podría utilizar posteriormente como el NodeMCU V2 o bien el microcontrolador ESP8266 junto con el módulo de Wifi *ESP12*.

Recordando la importancia de interpretar otras formas de controlar el aforo, comencé la investigación de la propuesta estudiada al inicio del proyecto. Por lo que, se estudió la idea de captar las presencia de personas a través de un sensor PIR. Este fue ajustado hasta capturar la presencia de un ser humano en un ritmo de movimiento real. Para ello se preciso entender la importancia del trigger asociado al sensor junto con mi compañero Víctor.

Posteriormente y siendo consciente que el código sobre TFL funcionaba, decidí comenzar a capturar el mapa de calor y así, poder completar un número suficiente de controles de aforo. Más tarde se implementó su lectura sobre la web gracias a la colaboración de Víctor y Miguel.

Una vez obtenida la serie de códigos asociados a la publicación de datos, procedí a mejorar el código lo máximo posibles hasta poder suplir el concepto de tiempo real, expuesto como objetivo. Durante las pruebas de código para la mejora, estuve detallando los procedimientos y códigos aplicados para el control de aforo con dos sensores PIR, a mi compañero Daniel.

Para finalizar quisiera hacer incapié a las secciones referentes a CV, las cuales fui encargado de redactar, y a la serie de pruebas asociadas a los *4* códigos de detección de E/S, junto con el mapa de calor.

Daniel Grado Guerrero

A lo largo de la evolución del proyecto he participado en la resolución de distintas tareas.

Al comienzo, fui el encargado de seguir desarrollando el modelo en el lenguaje de programación Python, de manera que a dicho código se le iban añadiendo nuevas características. Estas vienen a ser la modularización del código para que fuese más legible, organizado y, lo más importante, fácilmente modificable. Adicionalmente, se estableció una serie de comandos para que, en función del solicitado, se ejecutase el programa de una manera u otra.

Además, también realicé una interfaz gráfica simple para que se pudieran realizar ciertas tareas mediante unos clicks. Todas estas tareas se hicieron a modo de prueba en un principio, pero al incorporarse al modelo un broker con publicadores y suscriptores, dichas modificaciones carecían de sentido, por lo que fueron desestimadas.

Al finalizar el desarrollo sobre Python, me centré en ayudar a mi compañero Miguel

con la web, la cual sería de vital importancia, pues al fin y al cabo, es donde se muestran los datos recogidos y enviados. Colaboré en la realización de un fichero JSON, de manera que se establecieron una serie de parámetros configurables que pudieran ser modificados a posteriori por el usuario de una manera cómoda y sencilla.

De igual manera, entre Miguel y yo diseñamos y pusimos en práctica una serie de tablas en la base de datos, para así poder almacenar todos los que luego serían mostrados en la web.

Tras ello, fui el encargado de realizar y configurar los gráficos diario y semanal de la página de estadísticas, de manera que, gracias a Miguel por su dedicación con la persistencia de datos, se pudieran mostrar de una manera más amigable e intuitiva.

De la parte de web, la última aportación que realicé fue la de ayudar a que el servidor y la web pudieran desplegarse correctamente sobre una RPI. Esto se debe a que así, se podían abaratar los costes a futuro ya que en vez de necesitar un servidor propio más costoso o uno en la nube, se podía utilizar uno local. Miguel y Víctor fueron los encargados de realizar un primer despliegue de manera satisfactoria sobre un sistema operativo basado en Linux, y yo fui el que comprobó que efectivamente en una RPI también se podía realizar dicho despliegue.

En cuanto a los sensores, fui el encargado de realizar el modelado para el ultrasónico, basándome en el modelo ya realizado por Jesús y Víctor para el sensor PIR. Además de estos dos sensores, otros fueron estudiados con la idea de disponer de un catálogo más amplio y no depender de solo estos. A pesar de ello, no fue posible porque no eran óptimos para el correcto desempeño y funcionamiento del modelo establecido.

Tras haber completado de manera satisfactoria el modelado de los sensores, estuve realizando las pruebas para verificar el comportamiento de los sensores ante distintas situaciones a las que se enfrentarían.

Para finalizar, colaboré con las pruebas generales, a la vez que ayudé a realizar la memoria de este proyecto.

Victor Tello Carrascal

Durante el desarrollo de este proyecto he desempeñado diversas tareas:

Búsqueda y pruebas de diferentes códigos de los cuales partiría nuestro proyecto. Estos códigos estaban en su mayoría hechos en Python y debían cumplir el requisito de que funcionasen en hardware de bajo coste.

Una vez habíamos encontrado un código base, el siguiente paso fue comprenderlo y buscar soluciones para aumentar su rendimiento. Una de estas opciones fue realizar

un cambio de lenguaje a uno de más bajo nivel con el que obtuviéramos mejores resultados.

Colaboré con Jesús en el desarrollo del código para C++, este código tenía como principal objetivo mejorar al anterior escrito en Python. Para poder desarrollar y probar este código, hubo que buscar, compilar e instalar las distintas librerías de las cuales hacíamos uso.

Tras realizar el cambio de código nos dimos cuenta que una de las funciones consumía casi todo el tiempo de ejecución, por lo que procedí a la búsqueda de información y a la posterior reimplementación de estos códigos, comprobando su funcionamiento y observando que pese a ganar cierto rendimiento, la pérdida en cuanto a detección era mucho mayor.

La siguiente tarea que realicé fue la de comenzar a recopilar información sobre las formas de conectar las diferentes aplicaciones mediante IoT. De esta búsqueda de información salieron varias opciones, ya que, al ser un campo completamente nuevo, no sabía cuál de ellas era la más adecuada para este proyecto. Es por esto que, antes de desarrollar los códigos finales con la librería de Mosquitto, desarrollé diferentes pruebas con otras librerías para probar su funcionamiento y ver de primera mano las ventajas y las desventajas.

El siguiente paso fue desarrollar los diferentes códigos correspondientes a cada una de las aplicaciones. Primero de todo, se desarrolló el código de C++, ya que era el más sencillo de implementar para mi. Tras este, se creó el código de la parte de suscriptor de la página web. El primer código que cree no era fácilmente escalable, es por ello, que tuve que volver a implementarlo teniendo en cuenta esta característica. Al finalizar el desarrollo de estos módulos, se incluyeron en sendas aplicaciones.

Para poder comprobar su correcto funcionamiento, tuve que configurar el broker estableciendo un usuario y una contraseña para rechazar las posibles conexiones de dispositivos no deseados. Debido a las restricciones provocadas por la pandemia, tuvimos que pensar una solución (al no encontrarnos en la misma red) para el envío de los datos. Optamos por utilizar una VPN.

Una vez funcionaron los módulos, solucioné unos errores y procedí con la incorporación del envío de datos por parte del HeatMap. Primero de todo logramos enviar una matriz de datos y tras varios intentos, logré enviar una imagen sobre la cual pintar este mapa.

Debido principalmente a la cantidad de aplicaciones que teníamos, y la idea de, en un futuro hacer una aplicación escalable que fuera capaz de admitir varias zonas de

detección, reestructuré los temas de todas las aplicaciones y además creé una cola de mensajes en la aplicación web para que, en caso de recibir una gran cantidad de datos, la web fuese capaz de procesarlos sin perder ninguno.

Simultáneamente a esto, colaboré con Miguel y Daniel a buscar un fallo que ocurría a la hora de desplegar la aplicación web en Linux y que nos impedía montar un servidor en este sistema operativo.

Una vez desarrollado todo, tomé parte en la creación del código que se encargaba de detectar a personas mediante el sensor PIR y en la búsqueda de información de los diferentes sensores que se utilizarían posteriormente.

Finalmente, ayudé a hacer las diversas pruebas generales y de aplicación web así como en la documentación y posterior redacción de la memoria del proyecto.

Apéndice D

Código de cada programa

El código de cada uno de los programas desarrollados para las diferentes aplicaciones que componen este proyecto se encuentra alojado en el siguiente repositorio de GitHub: https://github.com/martellucm/TFG-Occupancy-2020-21.

Además, en este repositorio se encuentran las instrucciones de instalación de todos y cada uno de los programas y librerías necesarios para su correcto funcionamiento.

Autores del TFG
Víctor Tello Carrascal
Miguel Artell Moreno
Daniel Grado Guerrero
Jesús Álvarez Coll
FECHA
Ult. actualización 15 de junio de 2021

Autor de la plantilla
Contacto
Autor: David Pacios Izquiero
Correo: dpacios@ucm.es
ASCII: ascii@ucm.es
Despacho 110 - Facultad de Informática